Spécification formelle de systèmes temps réel répartis par une approche flots de données à contraintes temporelles

Tanguy Le Berre

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *Institut National Polytechnique de Toulouse*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Tanguy Le Berre*
Le *23 mars 2010*

Titre :
Spécification formelle de systèmes temps réel répartis par une approche flots de données à contraintes temporelles

---

JURY
Jean-Philippe Babau (PR) - LISyC, Brest
Michel Charpentier (PR) - UNH, Durham
Mamoun Filali Amine (CR) - IRIT, Toulouse
Dominique Méry (PR) - LORIA, Nancy
Olivier Roux (PR) - IRCCyN, Nantes
Gérard Padiou (PR) - IRIT, Toulouse
Philippe Quéinnec (MC) - IRIT, Toulouse

Ecole doctorale : *Mathématiques, Informatique et Télécommunications de Toulouse*
Unité de recherche : *Institut de Recherche en Informatique de Toulouse*
Directeur(s) de Thèse : *Gérard Padiou, Philippe Quéinnec*
Rapporteurs : *Jean-Philippe Babau, Michel Charpentier, Olivier Roux*

ABSTRACT

Real time systems are usually defined as systems where the total correctness of an operation depends not only on its logical correctness, but also on the execution time. Under this definition, time constraints are defined according to system operations. Another definition of real time systems is centered on data where the correctness of a system depends on the timed correctness of its data and of the data flows across the system. i.e. we expect the values taken by the variable to be regularly renewed and to be consistent with the environment and the other variables.

I propose a modeling framework based on this later definition. This approach allows users to focus on specifying time constraints attached to data and to postpone task and communication scheduling matters. The timed requirements are not expressed as constraints on the implantation mechanism, but on the relations binding the system's variables. These relations between data are expressed in terms of a so called observation relation which abstracts the relation between the values that are taken by some variables, the set of sources and the image. This relation abstracts the communication as well as the computational operations and a set of observation relations models the system architecture and the data flows by defining the paths along which values of sources are propagated to build the values of an image.

The real time properties are expressed as constraints on the propagation paths and state the temporal validity of the values. This temporal validity is defined by the time shift between the source and the image, and specifies the propagation of timely sound values along the path to build temporally correct values of the system outputs. At this level of abstraction, the designer gives a specification of the system based on timed properties about the timeline of data such as their freshness, stability, latency etc.

In order to prove the feasibility of an observation-based model, a finite state transition system bi-similar with the specification is built. The existence of a finite bi-similar system is deduced from the bounded time shift between the variables. The existence of an infinite execution in this system proves the feasibility of the specification.

RÉSUMÉ

Une définition des systèmes temps réel est que leur correction dépend de la correction fonctionnelle mais aussi du temps d'exécution des différentes opérations. Les propriétés temps réels sont alors exprimées comme des contraintes temporelles sur les opérations du système. Nous proposons dans cette thèse un autre point de vue où la correction est définie relativement à la validité temporelle des valeurs prises par les variables du système et aux flots de données qui parcourent le système.

Pour définir ces conditions de validité, nous nous intéressons au rythme de mise à jour des variables mais aussi aux liens entre les valeurs des différentes variables du système. Une relation dite d'observation est utilisée pour modéliser les communications et les calculs du système qui définissent les liens entre les variables. Un ensemble de relations d'observation modélise l'architecture et les flots de données du système en décrivant les chemins de propagation des valeurs dans le système.

Les propriétés temps réels sont alors exprimées comme des contraintes sur ces chemins de propagation permettant d'assurer la validité temporelle des valeurs prises par les variables. La validité temporelle d'une valeur est définie selon la validité temporelle des valeurs des autres variables dont elle dépend et selon le décalage temporel logique ou événementiel introduit par les communications ou les calculs le long des chemins de propagation.

Afin de prouver la satisfiabilité d'une spécification définie par une telle architecture et de telles propriétés, nous construisons un système de transitions à état fini bisimilaire à la spécification. L'existence de ce système fini est justifiée par des bornes sur le décalage temporel entre les variables du système. Il est alors possible d'explorer les exécutions définies par ce système de transitions afin de prouver l'existence d'exécutions infinies satisfaisant la spécification.

REMERCIEMENTS

Je souhaite ici remercier les personnes qui ont contribué, directement ou indirectement, à la réalisation de ma thèse et à la rédaction de ce mémoire.

Je remercie tout d'abord les rapporteurs de mon travail, Jean-Philippe Babau, Olivier Roux et Michel Charpentier. Je les remercie d'avoir pris le temps de lire le manuscrit, d'avoir commenté mon travail et m'avoir conseillé. Je remercie également les membres du jury Dominique Méry et Mamoun Filali de s'être intéressés à mon travail et d'être venus assister à ma soutenance.

Je tiens à remercier Gérard Padiou, mon directeur de thèse, pour m'avoir accueilli en Master de Recherche et m'avoir par la suite confié ce sujet de thèse. Malgré un emploi du temps chargé, il m'a accompagné tout au long de ces quatre ans et m'a proposé tout ce que l'on peut attendre d'un directeur de thèse, une oreille disponible et des conseils avisés.

Je remercie tout particulièrement Philippe Quéinnec, mon co-directeur de thèse. C'est véritablement une chance qu'il m'ait accompagné depuis quatre ans. Il s'est montré disponible afin de pouvoir débattre, parfois longuement, de mon travail. Lors de ces discussions, il a su me conseiller et m'encourager tout en me laissant libre des orientations que je voulais donner à ce travail. Sa rigueur m'a permis de corriger et perfectionner tout ce que j'ai pu effectuer et notamment ce manuscrit. Cette thèse ne serait certainement pas aussi aboutie s'il n'avait pas été là. Je m'excuse auprès de lui pour mes nombreuses fautes de grammaire ou d'orthographe qu'il a corrigées depuis quatre ans.

Je souhaite remercier l'ensemble de l'équipe de recherche et d'enseignement qui m'a accueilli et toutes les personnes que j'ai pu y côtoyer. Merci pour le cadre de travail que vous avez proposé, merci aux permanents qui m'ont aidé dans mes activités de recherche et d'enseignement et merci aux autres doctorants de l'équipe pour leur soutien mutuel. Merci pour la bonne humeur dont tout le monde fait preuve.

Je remercie mon entourage, à commencer par ma famille et en particulier mes parents qui m'ont soutenu et m'ont donné l'opportunité d'effectuer les études que je souhaitais et donc finalement d'entreprendre cette thèse. Je remercie mes amis, à Toulouse et ailleurs, qui ont suivi mon travail, m'ont encouragé et parfois m'ont détourné de la thèse en me proposant les distractions et les vacances nécessaires à mon équilibre. Je remercie finalement Perrine pour être à mes cotés en permanence, dans les bons comme dans les mauvais moments. Il m'est impossible d'imaginer avoir conclu ce travail sans ton support au quotidien. Merci pour tout.

Enfin, j'ai une dernière pensée pour la ville de Toulouse et le sud-ouest que je vais quitter. Sans exagérer comme mes amis toulousains et affirmer que Toulouse est la plus belle ville du monde, j'ai beaucoup apprécié la vie toulousaine. Ce fut un grand plaisir d'effectuer ma thèse ici et je reviendrai toujours avec plaisir.

TABLE DES MATIÈRES

1 INTRODUCTION

1.1 CONTEXTE

L'informatique fait désormais partie de notre quotidien. En effet de plus en plus de systèmes utilisés dans la vie quotidienne intègrent désormais des composants informatiques. Cette intégration est motivée par l'aide apportée par ces systèmes. En effet ces systèmes effectuent ou simplifient une partie des tâches dont nous devions nous charger. On cite par exemple l'intégration de l'informatique dans l'électroménager, l'audiovisuel ou les transports. Dans un système tel qu'une automobile, au delà de la simplification de la conduite et de notre confort, le système informatique s'assure aussi de notre sécurité. De par les enjeux humains, un dysfonctionnement est inacceptable et il faut donc pouvoir s'assurer que le système informatique remplit ses objectifs en toutes circonstances.

Vérifier la validité d'un système c'est s'assurer que le système ne commet pas d'erreurs. Afin d'assurer cette validité, il faut premièrement définir ce qu'est une erreur, ou à l'inverse définir par une spécification ce qu'est un comportement correct du système. Cette définition doit être formalisée, il ne doit pas y avoir d'ambiguïtés. On vérifie alors que le système respecte cette spécification. On peut raisonner sur une description du système et pas directement sur le système afin de prouver sa validité avant sa réalisation. Cela permet notamment de garantir que le système final qui est développé sera correct dès les premières étapes du développement. La description du système doit elle aussi être formalisée.

S'assurer que le système ne commet pas d'erreur signifie généralement que, quelles que soient les conditions dans lesquelles un calcul s'opère, son résultat est toujours correct. Il s'agit de l'aspect fonctionnel. On s'intéresse dans cette thèse à un autre aspect de la correction et à un type de système particulier, l'aspect temporel et les systèmes temps réel. Il s'agit généralement de systèmes qui, à partir de mesures caractérisant l'état de l'environnement, en déduisent une action à produire. L'environnement étant en évolution en permanence, la validité de la réaction est limitée dans le temps. Pour de tels systèmes, la validité s'exprime comme la justesse des calculs mais aussi par des propriétés temporelles. Généralement on s'assure de la correction temporelle en vérifiant que les opérations du système vérifient leurs échéances.

La validité de ces propriétés temps réel dépend des propriétés du système et notamment de son architecture. On s'intéresse plus particulièrement au cadre des systèmes distribués. Un tel système est réparti sur plusieurs calculateurs distincts et chaque information n'est pas directement disponible à l'ensemble du système. Des communications sont nécessaires mais elles complexifient le système en introduisant un décalage entre les différents sites, décalage éventuellement indéterministe. Le temps pris par la propagation des valeurs dans le système peut dont interférer avec certaines propriétés temporelles et complexifier leur vérification.

1.2 PROBLÉMATIQUE

On cherche dans cette thèse à formuler les propriétés temps réel d'un système par un ensemble de propriétés sur les données du système et non pas sur les opérations du système. On postule alors que les propriétés temporelles du système sont vérifiées si chaque donnée évolue conformément aux propriétés qui lui sont associées. On donne tout d'abord des propriétés propres à chaque variable, propriétés qui caractérisent la fréquence à laquelle une variable doit être mise à jour pour que ses valeurs soient valides. Dans le cas d'un système réactif, idéalement une variable du système doit évoluer au même rythme que l'environnement. Si cet environnement évolue de manière continue, ce n'est alors pas possible. Il s'agit donc de donner un rythme d'évolution acceptable de cette variable pour que ces valeurs soient temporellement valides.

Ensuite, la valeur d'une variable peut être obtenue à partir des valeurs d'autres variables. Un composant calcule la valeur de la sortie selon les valeurs des entrées qui ont été propagées dans le système, un système réactif décide de son action en fonction des mesures de l'environnement. La validité temporelle d'une valeur dépend alors des valeurs utilisées en entrée. On cherche par exemple à utiliser des valeurs fraîches en entrée. On veut donc exprimer la validité temporelle des valeurs prises par une variable selon la validité temporelle des valeurs d'autres variables progagées dans le système et donc selon les flots de données d'une variable vers une autre.

En exprimant les propriétés temporelles par rapport aux données, on cherche à se démarquer des analyses temps réel classiques où les propriétés temps réel sont exprimées par rapport aux mécanismes utilisés pour la réalisation du système et où la vérification est réalisée par une analyse d'ordonnancement. On veut exprimer les propriétés comme des attentes sur le comportement des données et non pas comme des attentes sur les mécanismes du système. Les mécanismes utilisés doivent cependant garantir les propriétés temporelles des données qu'ils utilisent et produisent.

Une deuxième problématique découle de la première. Une fois les propriétés exprimées, une fois que l'on a la spécification d'un système, il faut être capable d'analyser cette spécification. On veut vérifier s'il est possible de réaliser un système satisfaisant ces propriétés et vérifier la validité d'une solution proposée. Il faut prouver que de telles analyses sont possibles et déterminer s'il est possible de les automatiser.

1.3 TRAVAUX DE LA THÈSE

Nos travaux définissent un ensemble de propriétés utilisées pour exprimer les exigences temporelles sur un système. Afin de définir ces propriétés temporelles, on définit d'abord un ensemble de relations utilisées pour caractériser les liens entre les valeurs des différentes variables d'un système. Ce lien est exprimé par un ensemble de relations d'observation qui définissent l'architecture du système et modélisent les communications et calculs de ce système. La définition de cette relation s'abstrait des mécanismes utilisés. Elle définit uniquement un lien entre les valeurs d'une variable image et les valeurs dans un état précédent d'un ensemble de variables sources. On s'intéresse alors aux flots de données dans le système, c'est-à-dire à comment la valeur prise par une variable est propagée le long des différentes relations d'observation afin de produire les valeurs d'une autre variable. Un ensemble de relations d'observation construit alors des chemins de propagation.

Les propriétés temporelles du système sont exprimées par des propriétés temporelles sur ces chemins de propagation. Il s'agit par exemple de spécifier qu'une valeur en

sortie d'un calcul est temporellement valide uniquement si les valeurs utilisées en entrée vérifient elles-mêmes certaines propriétés temporelles telles que la stabilité ou la fraîcheur. On cherche donc à s'assurer qu'une valeur propagée dans le système est toujours temporellement valide lorsqu'elle est utilisée pour produire la valeur d'une autre variable. Ces propriétés contraignent le comportement temporel et les rythmes de mise à jour de chaque variable du système et lient le comportement temporel des différentes variables le long d'un chemin.

La spécification d'un système est donc définie comme un couple (architecture, propriétés temporelles). L'architecture définit les liens entre les valeurs des variables du système et les chemins de propagation des valeurs de chaque variable dans le système. Les propriétés temporelles contraignent les variables et les chemins de propagation du système afin de définir des dépendances entre la validité temporelle des valeurs des différentes variables.

On montre alors comment automatiser la vérification d'une telle spécification. Pour cela, on définit un système de transitions fini équivalent à la spécification. On utilise ce système de transitions pour prouver la satisfiabilité d'une spécification.

Les travaux donnés dans cette thèse permettent d'exprimer des propriétés temps réel sur les valeurs des variables d'un système, les liens entre les valeurs des différentes variables et les chemins de propagation de ces valeurs. Ces propriétés définissent alors les conditions de validité temporelle de ces valeurs en tout état du système. Il s'agit de propriétés ayant une sémantique originale, exprimées indépendamment des implantations des opérations du système et se différenciant des propriétés temps réel classiques exprimées comme un ensemble d'échéances. On donne de plus les moyens de vérifier que de telles propriétés définissent bien une spécification satisfiable.

1.4 PLAN

Le deuxième chapitre permet de détailler le contexte de ce manuscrit. On s'attache à illustrer et définir ce qu'est un système temps réel et quelles sont les propriétés que l'on veut spécifier. On motive la nécessité de s'assurer de la validité d'un tel système ainsi que les difficultés à surmonter pour pouvoir s'assurer de cette validité. On donne alors les solutions existantes pour spécifier un système, définir les propriétés temps réel qu'un système doit satisfaire, et décrire le comportement d'un système. Pour une spécification, on s'intéresse aux techniques permettant de s'assurer que le comportement modélisé vérifie bien les propriétés attendues. Enfin, on précise les objectifs de cette thèse par rapport aux travaux existants.

Dans le troisième chapitre, on définit formellement les relations entre les variables d'un système. Ces relations permettent de décrire l'architecture d'un système et de spécifier comment les valeurs de chaque variable se propagent dans le système. On introduit de plus les éléments permettant de caractériser le comportement temporel de chaque variable, c'est-à-dire les instants où chaque valeur de la variable est mise à jour et la notion d'occurrence.

Le quatrième chapitre définit des propriétés sur le comportement temporel des variables. On s'intéresse notamment aux chemins de propagation des valeurs des variables et aux liens entre les comportements temporels des différentes variables. Il s'agit de restreindre le décalage introduit le long des chemins de propagation et aussi de caractériser les intervalles de temps pendant lesquels une valeur propagée est valide. La spécification d'un système est alors définie par un ensemble de relations définissant

l'architecture du système et un ensemble de propriétés temporelles sur la propagation des valeurs dans cette architecture.

Le cinquième chapitre soulève le problème de la vérification des propriétés temporelles dans les premiers états d'une spécification. Un régime initial est défini dans lequel certaines propriétés ne peuvent pas être vérifiées. On montre que pour certaines spécifications, la sortie de ce régime se fait en un temps borné calculable.

Le sixième chapitre donne la définition d'un système de transitions représentant l'ensemble des exécutions définies par une spécification. Il s'agit de pouvoir construire les états du système définis par une telle spécification. Le chapitre suivant montre que ce système de transitions peut se ramener à un système de transitions à états finis.

Le huitième chapitre donne un exemple illustrant les chapitres précédents.

2 APPROCHES EXISTANTES

2.1 INTRODUCTION

2.1.1 *Systèmes temps réel*

Un système informatique valide est un système qui remplit son contrat et donc n'a pas un comportement autre que celui attendu. Dans le cas d'un système effectuant des calculs numériques, le contrat stipule notamment que les résultats des calculs doivent être justes. Dans [Sta88], une définition des systèmes temps réel est donné en référence à la validité d'un système informatique :

> In real-time computing the correctness of the system depends not only on the logical results but also on the time at which the results are produced.

En plus d'une validité du système classique liée à la justesse des calculs effectués, des propriétés de validité liées au temps sont introduites.

Ces systèmes sont généralement des systèmes réactifs dont le comportement dépend de l'environnement. On a alors un ou plusieurs capteurs permettant d'obtenir une approximation de l'environnement et de son évolution. Le système, selon cette approximation, décide de l'action ou plutôt de la réaction qu'il doit entreprendre. Une réaction peut ne plus être adaptée après une évolution de l'état de l'environnement et donc être valide pendant un temps limité. Il faut alors suivre les évolutions de l'environnement pour réadapter la réaction. Généralement, l'environnement est en évolution continue et ne pouvant avoir une mesure continue de l'état de l'environnement, on définit alors un rythme de fonctionnement des capteurs donnant un échantillonnage raisonnable de cette évolution. De même, le calcul de la réaction ne pouvant se faire en temps nul, on donne alors une échéance à ce temps de calcul. Le but est que malgré le temps pris par le calcul de la réaction, cette réaction reste cohérente par rapport à l'état de l'environnement.

Les définitions des propriétés temps réel sont alors généralement données à travers un temps de réponse du système, le temps maximum pour produire la réaction du système. De même que les calculs effectués par un système doivent être justes pour produire des résultats justes, on préfère formuler les propriétés temps réel sous une autre forme. S'il existe une échéance pour ce temps de calcul, c'est que le temps de réaction du système n'est pas nul et que la mesure de l'environnement en entrée n'est valide que pendant un certain intervalle de temps. Tout résultat d'un calcul effectué à partir d'une mesure de l'environnement est donc valide pendant un intervalle de temps d'autant plus limité que le temps de calcul est long. Les limites sur le temps de réponse du système dépendent en fait de la validité du résultat des calculs dans le temps. La définition d'un système temps réel sur laquelle repose les travaux de cette thèse est la suivante :

> La validité d'un système temps réel dépend de la validité temporelle de ses données.

Par validité temporelle des données, on entend que les différentes valeurs prises par les variables du système sont temporellement valides. Un exemple de validité temporelle

est d'exiger qu'une donnée soit fraîche et donc qu'elle soit mise à jour fréquemment. Une valeur peut aussi être définie comme temporellement valide uniquement si elle est basée sur des valeurs temporellement valides d'autres variables. Pour la valeur d'une variable, cette validité temporelle dépend alors de la sémantique de la variable, de la signification des différentes valeurs qu'elle prend (par exemple une température, une vitesse, une position, etc), et des liens avec les autres variables du système.

La validité d'un système ayant de telles propriétés dépend bien sûr des mécanismes qui réalisent ce système. Il s'agit cependant d'exprimer ces propriétés indépendamment des mécanismes. Une définition formelle de la validité temporelle de la valeur d'une variable est donnée par la suite.

2.1.2 *Contexte*

Les systèmes temps réel auxquels on s'intéresse sont des systèmes logiciels. Si, pour les exemples d'applications que nous donnons, de nombreuses fonctionnalités peuvent être assurées par des puces dédiées, désormais on se concentre plus sur la réalisation de ces fonctionnalités par des moyens logiciels.

On détermine deux types de systèmes temps réel selon l'importance des propriétés temps réel. Dans les systèmes temps réel durs, aucun dépassement des échéances n'est toléré contrairement aux systèmes temps réel souples. Les systèmes temps réel durs sont généralement des systèmes critiques. Compte tenu des enjeux présents dans les systèmes critiques, il est crucial de s'assurer qu'il n'y a pas d'erreur, et notamment d'erreur temporelle. Par exemple les systèmes de contrôle dans les centrales nucléaires ou les logiciels embarqués dans l'avionique ou l'automobile sont des systèmes critiques où les contraintes de temps sont importantes. Certains systèmes boursiers, sans qu'il y ait de tels enjeux, gèrent eux énormément d'argent et sont eux aussi des systèmes temps réel durs. Pour d'autres systèmes tel qu'un système GPS, un téléphone portable ou encore un jeu vidéo, une mauvaise réactivité et un non respect des contraintes temporelles entraîne une gêne et nuit au plaisir de l'utilisateur. On ne demande alors pas à avoir une correction parfaite du système mais à assurer une qualité minimum de service. Il suffit que la plupart des traitements soient effectués dans les temps, suffisamment souvent afin d'assurer le plaisir de l'utilisateur. Ce sont alors des systèmes temps réel souples. On se concentre dans cette thèse sur les systèmes temps réel durs pour lesquelles les exigences temporelles sont critiques. On vérifie donc que nos exigences sont toujours respectées.

Vérifier que nos exigences sont toujours respectées n'est pas un objectif propre au domaine temps réel :

> Building systems which are correct with respect to given requirements is the main challenge for all engineering disciplines. [Sifakis - Présentation à l'Embedded Systems Week 2008]

On parle de défi (challenge) car il existe des motivations claires pour s'assurer de la correction du système mais, comme pour d'autres problèmes, s'il existe des solutions simples pour de petits systèmes, cela se complique pour des systèmes réellement utilisés et de taille plus importante. On définit les principaux obstacles qui peuvent se poser pour atteindre cet objectif. On se concentre principalement sur la correction temporelle même si ces obstacles sont généralement les mêmes pour s'assurer d'autres types de correction.

Le principal problème est que les systèmes informatiques, de manière générale, deviennent de plus en plus complexes. Que ce soit le nombre de lignes de codes ou le

nombre d'interactions entre les différents éléments, toutes les mesures de complexité ont augmenté et continuent d'augmenter. On demande notamment à chaque système de remplir de plus en plus de fonctionnalités, de plus en plus vite, dans des environnements de plus en plus variés. Cette augmentation de la complexité rend encore plus difficile une analyse permettant de garantir les propriétés du système, parmi lesquelles les propriétés temps réel jouent un rôle important. Lorsque l'on est capable d'analyser un système, il faut alors pouvoir passer à l'échelle et adapter les techniques d'analyse à des systèmes plus complexes.

Un des aspects de cette complexité est la répartition des systèmes. L'architecture du système est de plus en plus souvent répartie entraînant des communications par message entre les différents constituants du système. Ces échanges peuvent être réalisés à travers un bus de terrain (pour l'automobile par exemple) ou à travers internet (par exemple le multimédia ou les jeux vidéos). Dans un contexte temps réel, il faut gérer le décalage introduit entre les différents éléments du système. Il faut alors concevoir des protocoles de communication permettant de déterminiser le déroulement des communications ou permettant de prévoir les différents comportements possibles. Un réseau totalement imprévisible ne peut pas garantir les exigences dures d'un système temps réel et peut difficilement assurer la qualité de service demandée pour un système temps réel souple.

On demande de plus aux systèmes de pouvoir être déployés dans plusieurs environnements ou sur des plateformes différentes. Selon ces plateformes, le comportement, notamment temporel, du système peut changer. Par exemple, le temps d'exécution d'une séquence de code dépend des propriétés de la plateforme matérielle. On veut donc avoir un système qui ne dépend pas ou pas trop de cette plateforme et donc on cherche à paramétriser la dépendance du logiciel vis-à-vis de son environnement d'exécution. Une possibilité est de pouvoir facilement modifier le système selon la plateforme utilisée. Pour cela, les techniques qui permettent de s'assurer de sa correction doivent être robustes. C'est-à-dire qu'en cas de modification d'une partie du système ou de l'environnement d'exécution du système, il est possible de réutiliser les résultats des analyses précédentes. Les modifications et ainsi l'adaptation du système sont alors plus rapides.

La plupart des analyses effectuées dans le cadre des systèmes temps réel se basent sur des calculs pessimistes. De telles analyses sont plus rapides ou plus simples qu'une analyse plus fine mais une analyse non concluante n'implique alors pas nécessairement la non correction du système puisqu'elle se fonde sur une approximation du système. Un défi est alors de trouver de nouvelles méthodes d'analyse, ou encore d'améliorer celles existantes afin d'avoir des résultats moins pessimistes.

Finalement, un défi évident est de minimiser les coûts de développement d'un système temps réel tout en répondant aux autres défis. Tous ces défis ne peuvent être relevés que par l'élaboration d'un cycle de développement prenant en compte les exigences temps réel dès les premières étapes.

2.1.3 *Méthodes de vérification*

Plusieurs méthodes existent pour vérifier les propriétés d'une spécification. On donne les principales méthodes employées. Le but de ces méthodes est de découvrir des erreurs lors du développement et les diagnostiquer pour pouvoir les corriger au plus tôt.

TEST. Une première approche est l'utilisation de procédures de test. Il s'agit de définir des cas d'utilisation du système, par exemple des ensembles d'entrées possibles, et

tester, analyser le comportement du système. On compare le comportement obtenu au comportement attendu. Une procédure de test est parfaite si elle confronte le système à l'ensemble des entrées possibles. Cette exhaustivité n'est généralement pas possible. Les tests ne permettent pas de valider totalement le système final. Il existe cependant des outils permettant de générer des jeux de tests à partir d'une spécification afin d'améliorer la couverture et donc l'efficacité de ces tests.

PREUVE. Une autre méthode consiste à démontrer la validité du système final en construisant une preuve basée sur des règles de déduction logique. On utilise alors des assistants de preuves tels que Coq [The08] permettant d'automatiser une partie de la preuve. Si la théorie utilisée est consistante et si la preuve est correcte vis-à-vis de celle-ci alors cette méthode permet de s'assurer de manière absolue de la validité du système. Cependant c'est une méthode dont la complexité augmente très vite avec la taille du système, qui demande des connaissances au développeur et qui est difficilement automatisable. Elle est pour ces raisons encore difficile à mettre en œuvre dans l'industrie.

MODEL CHECKING. Le model checking ou vérification par modèles est aussi une méthode formelle permettant de s'assurer de la validité du système. Il s'agit d'explorer l'ensemble des exécutions et des états du système. Si l'ensemble des états n'est pas fini, il faut pouvoir construire un système fini équivalent au système en utilisant une représentation abstraite des différents états et des séquences d'états. Une fois construite une représentation finie de l'ensemble des états, on vérifie les propriétés de la spécification dans tous ces états et dans toutes les séquences d'états. En pratique il faut de plus que l'ensemble des états ou leur représentation soit de taille non seulement finie mais aussi raisonnable.

GÉNÉRATION AUTOMATIQUE. Une dernière méthode est de s'appuyer sur des outils de génération automatique d'un système respectant la spécification. Il s'agit d'une preuve par construction. On se concentre sur les propriétés de la technique de génération plutôt que sur le système généré. On prouve préalablement que l'outil de génération conserve les propriétés d'un ensemble de spécifications, notamment par l'utilisation de méthodes de preuve. Par la suite, pour toute spécification de cet ensemble on génère automatiquement une implantation. On a donc un travail préliminaire important mais qui amène à la possibilité d'utiliser un outil puissant et simple.

AUTRES TECHNIQUES. Il existe d'autres méthodes de vérification tel que l'interprétation abstraite ou les techniques de vérification en ligne des programmes. De plus, il existe d'autres méthodes spécifiques au langage de spécification utilisé et/ou aux propriétés à vérifier.

2.1.4 *Objet de l'étude*

On cherche à donner une spécification de systèmes temps réel dans un cadre formel avec une sémantique claire et permettant de procéder à une vérification, si possible automatique, du système. De plus, on cherche à exprimer ces propriétés temps réel sur les variables et les valeurs prises par ces variables et à définir des intervalles de validité temporelle pour ces valeurs. La définition de ces intervalles dépend de la sémantique des variables mais aussi des relations entre les variables. Pour les relations entre les

variables et donc entre les valeurs prises par ces variables, on veut spécifier des relations entre les validités temporelles de ces valeurs et des relations sur la propagation des valeurs dans le système.

Dans les sections suivantes, on donne les définitions, techniques et moyens les plus utilisés pour répondre à l'objet de notre étude : la spécification et l'analyse de systèmes temps réel. On s'intéresse à la définition des propriétés temps réel d'un système et notamment les propriétés sur les variables du système et leurs valeurs, à la sémantique de ces propriétés et aux moyens de vérification de ces propriétés.

Les propriétés temps réel décrivent dans quelles conditions doivent se dérouler les exécutions du système. On s'intéresse plus précisément à la construction de ces exécutions dans le cadre des systèmes de transition. On détaille donc les différents types de systèmes de transitions adaptés à la spécification de systèmes temps réel. Pour chacun de ces formalismes, on rappelle les techniques de vérification associées et les propriétés qu'il est possible de vérifier.

2.2 LOGIQUES TEMPORELLES

Les logiques temporelles sont utilisées pour spécifier des propriétés comportementales d'un système. Ce sont des logiques modales incluant la notion de temps. Une logique modale est une logique où des opérateurs sont introduits pour relativiser la vérité d'une proposition. Ces opérateurs sont comparables à des adverbes tels que peut être, sûrement, sans doute, ... Ils complètent une quantification plus précise. Une logique temporelle est une logique modale où les opérateurs introduits portent sur les instants pour lesquels les propriétés sont vérifiées. On peut exprimer des propriétés sur la validité d'une proposition dans le présent mais aussi le passé et le futur. Les opérateurs les plus courants sont les suivants :

- $\Box$ ou G : toujours, dans le futur la propriété sera toujours vraie
- $\Diamond$ ou F : finalement, dans le futur la propriété finira par être vraie
- A : pour toute exécution à partir de l'état courant la propriété sera vraie
- E : il existe une exécution à partir de l'état courant où la propriété sera vraie
- $\circ$ ou N (next) : la propriété sera vraie dans l'état suivant
- $\phi \mathcal{U} \psi$ (until) : ϕ est vrai jusqu'à ce que ψ le devienne

Une logique avec les opérateurs A et E est une logique arborescente. Ces logiques distinguent les différentes exécutions possibles. Un exemple de logique arborescente est CTL (Computational Tree Logic). Si ces opérateurs ne sont pas utilisés alors on a une logique linéaire, par exemple LTL (Linear Temporal Logic). Ces opérateurs sont définis par rapport au temps logique et ne permettent pas directement de définir une échéance temps réel.

CLASSIFICATION DES LOGIQUES TEMPORELLES. Il existe différentes logiques temporelles dont certaines sont définies pour la spécification de systèmes temps réel. [BMN00] donnent différents critères permettant de comparer ces logiques.

Tout d'abord, ces différentes logiques temporelles ne sont pas basées sur les mêmes logiques classiques. On peut se baser sur une logique propositionnelle, du premier ordre, ou encore du second ordre. La logique peut être déductive, c'est-à-dire qu'un ensemble de règles de déduction peut exister pour prouver l'implication entre deux propositions.

Les logiques temporelles utilisent une représentation du temps sous forme de points ou d'intervalles. Les opérateurs sont définis selon la représentation du temps utilisée.

Dans le cas de points, on s'attache alors à définir la précédence d'un événement par rapport à un autre événement. Si l'on utilise des intervalles, des opérateurs définissent si les intervalles se chevauchent, sont inclus, ...

Une mesure du temps peut être définie. Cette mesure peut être implicite ou explicite. Si elle est explicite, il est possible d'accéder à la valeur absolue du temps, de manipuler la date courante ou les dates futures. Si la mesure est implicite, on peut uniquement parler en temps relatif, dans 5 secondes par exemple. L'utilisation du temps explicite augmente l'expressivité et permet d'exprimer des propriétés telles que l'occurrence d'un événement quand la mesure de temps est paire. La mesure du temps peut être absolue et exprimée en secondes ou millisecondes, ou relative à une unité de temps non définie mais qui est définie par la suite lors de l'implantation.

Les logiques CTL et LTL n'offrent pas de mesure du temps et ne sont donc pas adaptées à la spécification de propriétés temps réel. Il existe cependant des extensions de ces logiques proposant une mesure du temps. Par exemple, RTCTL [EMSS91] étend l'opérateur $\mathcal{U}$ par une borne temporelle. Par exemple, la relation $\phi\, \mathcal{U}_{\sim k} \psi$ où $\sim \in \{\leq, =, \geq\}$ indique que ϕ est vrai jusqu'à ce que ψ le devienne et le temps écoulé avant que ψ soit vrai devra vérifier la contrainte par rapport aux k unités de temps. Il est possible de combiner les contraintes sur le temps écoulé pour exprimer des contraintes sur des intervalles ($k \in [1;3]$). On peut alors quantifier le temps écoulé entre deux événements et combien de temps une propriété reste vraie.

La décidabilité de ces logiques et notamment l'universalité et la satisfiabilité d'une formule sont étudiées. Les logiques sont décidables pour ces propriétés s'il existe une procédure finie permettant de prouver ou d'infirmer ces propriétés. Cette propriété assure la possibilité de créer des outils permettant l'analyse des formules d'une logique et accroît donc l'intérêt pour cette logique. On s'intéresse aussi à la complexité de la procédure de décision. La décidabilité d'une logique temporelle dépend de la mesure du temps utilisé mais surtout de la logique classique sur laquelle est basée la logique temporelle. Les logiques LTL et CTL sont décidables pour la satisfiabilité d'une formule, de même pour la logique RTCTL.

Les logiques temporelles sont utilisées pour spécifier les formules qu'une spécification opérationnelle doit respecter. Pour les systèmes de transitions décrits par la suite, de nombreux outils, notamment de model checking, permettent de vérifier le respect d'une formule spécifiée dans une logique temporelle adaptée.

Nous n'utilisons pas de logiques temporelles pour définir la sémantique des propriétés que l'on donne par la suite. De par leur expressivité, il est possible d'utiliser ces logiques pour donner des propriétés sur la propagation des valeurs dans le système. L'expression et la manipulation de ces propriétés est cependant difficile. Par conséquent les logiques temporelles ne paraissent pas un outil adapté pour décrire simplement la propagation des valeurs d'une variable dans le système et les propriétés temps réel de ces valeurs.

2.3 APPROCHES DÉDIÉES POUR LA SPÉCIFICATION DE SYSTÈMES TEMPS RÉELS

2.3.1 *Analyse d'ordonnancement*

Une approche classique aux problématiques temps réel est une analyse d'ordonnancement. Un processeur est une ressource où, à un instant donné, une seule tâche peut s'exécuter. On définit alors un ordonnanceur qui fixe en chaque instant quelle tâche s'exécute lorsque plusieurs veulent accéder au processeur ou à une autre ressource en

parallèle. Le choix de la tâche est fait selon une politique d'ordonnancement. On vérifie alors que cette politique d'ordonnancement permet d'assurer les propriétés temps réel du système.

Dans ce cadre, les propriétés temps réel d'un système sont exprimées comme un ensemble de caractéristiques des tâches :

- les instants d'activation, c'est-à-dire les instants auxquels une tâche demande l'accès au processeur pour débuter un calcul. On caractérise alors le rythme auquel les tâches sont effectuées en donnant les caractéristiques temporelles de ces instants d'activation. Ces activations peuvent être :
  - périodiques : tous les P unités de temps où P est la période ;
  - sporadiques : existence d'un temps minimum entre chaque activation ;
  - apériodique : impossibilité de caractériser le temps entre deux activations (une analyse est alors difficile).
- le temps de calcul de la tâche, combien de temps prend le calcul de cette tâche sur le processeur disponible ou des bornes minimum et maximum sur ce temps de calcul.
- les échéances : le temps accordé depuis l'instant d'activation pour exécuter la tâche.

La politique d'ordonnancement doit alors s'assurer, compte tenu des caractéristiques des tâches, que toutes les échéances vont être respectées. Il existe plusieurs politiques d'ordonnancement. La priorité des tâches peut être statiquement définie à l'avance ou dynamiquement selon l'évolution du système au cours de l'exécution. La politique d'ordonnancement doit aussi vérifier les accès aux ressources partagées critiques qui peuvent créer un interblocage.

Pour les politiques d'ordonnancement classiques, il existe des méthodes d'analyse permettant de vérifier si la politique d'ordonnancement choisie permet de respecter les échéances. Ainsi [LL73] présente un algorithme pour vérifier qu'une politique d'ordonnancement *Rate Monotonic* est correcte. Cette politique donne la plus haute priorité aux tâches ayant la période la plus faible. Une condition nécessaire basée sur un calcul de charge processeur est donnée. Une condition nécessaire et suffisante est basée sur le calcul du pire temps d'exécution de chaque tâche.

Certaines techniques d'analyse telles que celles données par [SRL90] prennent en compte les interactions entre les tâches et notamment les temps de blocage dus aux algorithmes gérant le partage de ressources entre plusieurs tâches. D'autres analyses telles que [TC94] prennent en compte la distribution du système et les communications entre tâches.

Une analyse d'ordonnancement est basée sur une spécification décrite par un ensemble de caractéristiques et exigences simples sur les tâches du système. On se concentre donc sur les propriétés des mécanismes et pas sur des propriétés de l'état et des valeurs des variables du système. On peut cependant penser que ces caractéristiques, telles que le rythme d'activation, sont données dans le but d'assurer un rythme de mise à jour correct des valeurs prises par les variables. Des propriétés données sur les données du système peuvent donc induire des propriétés des tâches du système pouvant être vérifiées par une analyse d'ordonnancement.

2.3.2 *Langages synchrones*

Les langages synchrones sont une famille de langages basées sur l'hypothèse synchrone. Cette hypothèse postule l'existence d'une échelle de temps logique discrète constituée d'instants. Chacun de ces instants correspond à une réaction du système. Les

événements qui déclenchent cette réaction sont simultanés et la réaction se fait en un temps nul. A chaque instant, donc, les entrées sont lues et les réactions qui en découlent sont produites de manière instantanée.

Cette hypothèse permet de simplifier la spécification des systèmes réactifs. Ce sont des langages expressifs spécialisés dans l'expression rigoureuse des réactions, ceci tout en gardant une sémantique simple et intuitive. Ils permettent de spécifier des comportements déterministes même en présence de parallélisme et de communication. Le compilateur assure ce déterminisme. L'utilisation de langages synchrones, de par ce déterminisme, permet de s'assurer facilement qu'une implantation vérifie une spécification. Un langage synchrone n'est alors pas un moyen d'exprimer des propriétés temps réel mais une possibilité pour la réalisation de ces propriétés, un langage d'implantation.

Lustre [HCRP91] et Esterel [BdS91] sont deux langages synchrones largement utilisés dans le milieu industriel, notamment l'aéronautique, grâce aux outils associés. Scade (Safety Critical Application Development Environment) est un environnement de développement basé sur le langage Lustre. Il permet de générer du code en langage C ou Ada. Lesar est par exemple un outil de vérification dédié au langage Lustre et Xeve est un environnement de vérification dédié au langage Esterel.

L'inconvénient majeur des langages synchrones est que le monde réel n'est pas synchrone mais fortement asynchrone. Par exemple, l'approche synchrone nécessite une synchronisation des horloges du système qui est difficile dans un système distribué. Lors d'un développement logiciel, certains sous-ensembles du logiciel sont développés avec une approche non-synchrone et l'intégration de composants synchrones avec des composants asynchrones peut alors invalider l'hypothèse synchrone. De plus, lors du développement le programmeur doit être conscient du cycle temporel utilisé afin de ne pas, par exemple, créer de boucle de causalité. Finalement, les langages synchrones sont souvent basés sur des primitives simples et n'offrent pas ou peu de possibilités pour gérer des structures de données complexes.

2.3.3 *Flot de données synchrones*

Lorsqu'on s'intéresse aux flots de données dans un système, on donne généralement une représentation de ce système comme un graphe orienté. Ce graphe représente la propagation des données du système et les nœuds représentent alors des unités de calcul. [BML99, Fon01] s'intéresse aux flots de données synchrones (synchronous data flow) . Dans ce cas, les arcs de ce graphe représentent des buffers FIFO stockant les données propagées. Les unités de calcul lors de leur activation consomment un certain nombre de données.

Il faut donc activer les calculs uniquement lorsque l'on est assuré de la disponibilité des entrées. Il s'agit alors de faire une analyse permettant de trouver un ordonnancement assurant cette disponibilité. Une solution synchrone est alors proposée.

Dans notre cas, les propriétés qui nous intéressent sont l'utilisation de valeurs temporellement valides en entrée. Ce n'est donc pas la quantité d'informations disponibles en entrée mais leur qualité qui est considérée. Il s'agit alors de trouver une implantation et par exemple un ordonnancement des calculs, assurant l'utilisation de valeurs temporellement valides.

2.3.4 *Suivi d'occurrences*

Dans [BJB05], les auteurs s'intéressent à la qualité de service et notamment aux propriétés temporelles des données propagées dans le système. Le système étudié est un système d'acquisition de données constitué d'un capteur physique mesurant un phénomène extérieur et produisant à chaque mesure une nouvelle occurrence, d'une application temps réel utilisant les occurrences produites par le capteur et d'une interface de communication et un driver permettant de faire le lien entre le capteur et l'application temps réel. L'article définit la qualité de service par rapport au cycle de vie des occurrences et s'intéresse donc au retard introduit entre l'instant de production des occurrences et leur utilisation par l'application temps réel. Il s'intéresse aussi aux pertes c'est-à-dire le nombre d'occurrences produites mais non utilisées par l'application. En faisant varier les propriétés temporelles des différents éléments du système d'acquisition et notamment leurs rythmes d'activation, des variations de la qualité de service sont observées. Ces variations motivent le suivi de la propagation des occurrences pour s'assurer de leur qualité temporelle lors de leur utilisation et donc de la qualité du système.

Afin d'étudier les propriétés des occurrences produites par le capteur en entrée du système, cette notion d'occurrence est formalisée dans [MBB09]. Chaque occurrence est représentée par un triplet constitué par une valeur, une date de production et un compteur dépendant de l'ordre de production. L'étude du cycle de vie des occurrences se réduit ici à l'étude de systèmes où les composants intermédiaires propageant les occurrences ont une seule entrée et une seule sortie. Ces composants retardent la propagation des occurrences et modifient leurs valeurs mais pas les autres éléments du triplet et notamment la date de production. Un observateur est utilisé pour suivre le parcours de chaque occurrence dans le système et déduire de la date de production de chaque occurrence le délai introduit par le système.

L'article introduit de plus les moyens de s'assurer que cette qualité de service répond aux exigences en effectuant une vérification utilisant des automates temporisés. La thèse [Ben08] formalise ce travail et propose utilisation de l'outil de modélisation et de vérification IF [BGL02].

La formalisation des occurrences est étendue dans [DB05] à des chaînes d'acquisition plus complexes avec des composants intermédiaires pouvant avoir plusieurs entrées. On a par exemple un composant calculant sa sortie selon des occurrences provenant de plusieurs capteurs physiques et une occurrence en sortie peut dépendre d'occurrences provenant de plusieurs sources. Les occurrences en sortie sont donc une fusion des occurrences en entrée d'un tel composant. Les caractéristiques temporelles telles que la date de production des occurrences en sortie sont alors définies en fonction des caractéristiques temporelles des entrées, par référence à un maximum ou à un minimum.

Dans notre travail, nous définissons nous aussi par la suite la notion d'occurrence d'une variable. Cette notion permet de compléter la notion de valeur pour décrire le comportement d'une variable. Les chemins de propagations définis ici diffèrent des chemins d'acquisition définis dans [Ben08] et [MBB09] car un composant peut comme dans l'article [DB05] utiliser en entrée des occurrences provenant de diverses sources. Cependant, on ne définit pas les caractéristiques temporelles des occurrences en sorties par rapport aux occurrences des sources en entrée. Le composant est ici considéré à la fois comme une source produisant les occurrences en sortie et comme un élément intermédiaire propageant les occurrences en entrée sans en modifier les caractéristiques temporelles. Finalement, on s'intéresse à un ensemble de propriétés temporelles plus

vaste que le retard défini dans [Ben08] et pouvant être défini entre n'importe chemin de propagation ou sous chemin de propagation du système et non pas uniquement entre producteur source et consommateur final. Il en résulte cependant dans notre travail une complexité accrue et une étude des propriétés plus difficile et a priori moins performante.

2.3.5 *Base de données temps réel*

Dans le domaine des bases de données, il est possible de donner des propriétés temps réel sur les transactions mais on s'intéresse aussi à la validité temporelle des valeurs stockées dans la base de données.

Dans [XSS+96], des intervalles de validités sont associées à chaque donnée de la base. Les échéances des transactions ne sont pas fixées à l'avance. Elles dépendent des données utilisées et de l'instant où ces données sont lues. En effet, une donnée lue par une transaction en fin de son intervalle de validité implique une échéance courte pour cette transaction. Il s'agit alors d'effectuer une analyse d'ordonnancement permettant de respecter ces échéances.

Dans [ABRW93], l'intervalle de validité d'une valeur est définie par rapport au décalage avec l'environnement. La valeur doit être une mesure de l'environnement dans un état récent. De même, pour le résultat d'un calcul, le décalage avec l'environnement autorisé pour ce résultat est le minimum des décalages autorisés pour les entrées du calcul. De ces intervalles de validité, un ordonnancement du système est déduit sous la forme d'un ensemble de transactions périodiques synchronisées.

Nous cherchons à exprimer des propriétés sur les valeurs semblables à celles qui viennent d'être présentées. On s'intéresse notamment, comme dans [ABRW93], aux liens entre les validités temporelles de chaque valeur des variables selon les liens entre les valeurs de ces variables. On souhaite cependant donner une sémantique plus riche à ces propriétés. La validité temporelle du résultat d'un calcul dépend des entrées mais dépend aussi de la signification de ce calcul. On peut ainsi avoir un décalage avec l'environnement plus important que celui autorisé pour les entrées d'un calcul selon les cas.

2.4 APPROCHES DIRIGÉES PAR LES MODÈLES

2.4.1 *UML*

UML (United Modeling Language) est un langage de modélisation graphique des systèmes. Un modèle est une représentation des caractéristiques du système et a pour objectif de structurer les données, les traitements et les flux d'informations entre les entités composant le système. Le langage UML est parmi les plus développés et utilisés des langages de modélisation. La modélisation d'un système en UML s'appuie sur plusieurs vues et plusieurs types de diagrammes. Les vues donnent l'organisation du système selon un certain point de vue. La vue des cas d'utilisations décrit des scénarios d'usage du système par les utilisateurs, la vue logique décrit comment sont satisfaits les besoins des acteurs (l'intérieur du système), etc... Les différents types de diagrammes sont utilisés pour définir chacune des vues. Il y a des diagrammes structurels décrivant l'architecture du système et des diagrammes comportementaux décrivant le comportement du système.

PROFIL UML. La sémantique d'UML telle que définie à l'origine ne permet pas ou peu de décrire des propriétés temps réel du système. Pour compléter la sémantique de base d'UML, on définit un profil UML. Plusieurs profils UML dédiés au temps réel existent, on cite notamment le profil MARTE (Modeling and Analysis of Real-time and Embedded systems) [AMdS07]. Ce profil a pour but de permettre la spécification et la vérification des propriétés temps réel d'un système. L'accent est mis sur la performance du système et doit permettre de réaliser des analyses d'ordonnancement et de limiter le temps écoulé entre deux événements. Les propriétés temps réel exprimées sont principalement des propriétés sur les tâches du système ou des caractéristiques de la plateforme matérielle du système, afin d'évaluer les performances du système.

D'autres approches permettent de spécifier d'autres propriétés. Par exemple [AF03] utilise le langage OCL pour spécifier pour chaque donnée du système un domaine de validité temporelle. En dehors de ce domaine, la valeur n'est plus valable. On vérifie alors que les événements qui utilisent cette valeur le font dans son domaine de validité. Notre but est cependant de s'assurer en permanence de la validité temporelle des données et non pas de définir cette validité lors des événements système.

UML permet de spécifier une large variété de systèmes, notamment par l'utilisation de profils adaptés au domaine cible. La sémantique n'est cependant pas suffisamment formalisée, et, si l'approche graphique permet une facilité d'utilisation, les ambiguïtés sémantiques ne sont pas compatibles avec une vérification complète du système. Les propriétés temps réel que l'on peut exprimer dépendent bien sûr du profil utilisé mais les propriétés les plus courantes sont celles utilisées pour effectuer une analyse d'ordonnancement, c'est-à-dire les caractéristiques et échéances pour les tâches du système.

2.4.2 *AADL*

AADL (Architecture Analysis and Design Language) [FGJ06] est un langage de description d'architecture destiné aux systèmes embarqués et donc qui s'intéresse aux problématiques temps réel. C'est un langage à la fois graphique comme UML mais aussi textuel. Il est basé sur la notion de composants qui définissent l'architecture du système. Ces composants sont hiérarchisés, composés et interconnectés. Il existe dix catégories de composants répartis en trois familles : les composants logiciels, les composants matériels et les composants composites. La communication entre composants est décrite à l'aide de ports et de connexions permettant plusieurs types de communications. AADL est particulièrement utilisé pour décrire l'architecture du système mais aussi le comportement de chacun des composants et des connexions entre composants. Des travaux existent pour formaliser la sémantique d'AADL, par exemple [BCFS05] et aussi pour permettre le développement de différents outils d'analyse et de génération de code. On citera notamment des outils d'analyse d'ordonnancement mais aussi des outils pour la vérification des propriétés de sûreté. Un profil UML existe pour décrire le langage AADL.

Il est possible d'ajouter des propriétés aux différents éléments composant un modèle et donc de compléter la sémantique d'AADL. Dans le cadre des systèmes temps réel, les analyses des propriétés temps réel qui sont possibles dépendent des outils qui existent. Si la sémantique AADL est plus formalisée que celle d'UML, on a, concernant les propriétés temps réel, un problème semblable à celui de l'utilisation d'UML. On est limité par les propriétés de base et les outils existants, même s'il est possible de compléter le modèle.

2.5 LES SYSTÈMES DE TRANSITIONS

Afin de donner la spécification d'un système, on s'intéresse aux systèmes de transitions et aux techniques de vérification de ces systèmes. On définit ce qu'est un système de transitions et les propriétés de ces systèmes que l'on cherche à vérifier. On cite ensuite les familles de systèmes de transitions les plus utilisées. Pour chaque type de système de transitions, on s'intéresse à l'utilisation qui en est faite, c'est-à-dire aux domaines dans lesquelles ils sont utilisés pour spécifier un système.

2.5.1 *Introduction*

Pour introduire les systèmes de transitions, on définit ce qu'est une trace, une exécution et tout d'abord un état.

Un état caractérise le système en un instant.

Définition . *État d'un système. Un état est défini par une affectation des valeurs à chaque variable du système. Pour une variable ν, on note $\nu.\sigma_i$ la valeur de ν dans l'état σ_i.*

L'ensemble des états est généralement défini comme un produit cartésien des domaines de définitions des variables du système.

Définition . *Trace. Une trace $\sigma_0, \sigma_1, \sigma_2, \ldots \sigma_i, \sigma_{i+1}, \ldots$ est une séquence finie ou infinie d'états.*

Définition . *Exécution d'un système. Une exécution d'un système est une séquence finie ou infinie d'états vérifiant les propriétés du système.*

Une exécution décrit un comportement possible du système. Il s'agit d'une séquence d'états donnant l'évolution des valeurs prises par les variables du le système au cours du temps. Pour l'analyse des systèmes dans un cadre formel, on s'intéresse à la sémantique opérationnelle de ces systèmes.

Définition . *Sémantique opérationnelle. La sémantique opérationnelle d'un formalisme permet de donner aux spécifications exprimées dans ce formalisme une interprétation sous la forme d'un ensemble d'exécutions.*

Au lieu de parler de l'interprétation d'une spécification dans le cadre de la sémantique opérationnelle d'un formalisme, on parlera de la sémantique opérationnelle de cette spécification. Cette interprétation sous la forme d'ensembles d'exécutions permet de décrire l'ensemble des comportements possibles d'un système. On note $[\![S]\!]$ la sémantique d'une spécification S donnée par la sémantique du formalisme associée à cette spécification.

Les systèmes de transitions sont un formalisme permettant de décrire l'évolution d'un système informatique dans le cadre du modèle de traces. C'est une machine abstraite construisant les séquences d'états possibles au cours d'une exécution. Il existe plusieurs types de système de transitions mais voici une définition générale.

Définition . *Système de transitions. Un système de transitions est défini par un triplet $(\Sigma, \Sigma_0, \rightarrow)$ avec :*

- *Σ : un ensemble d'états ;*
- *Σ_0 : l'ensemble des états initiaux ;*
- *$\rightarrow \subseteq \Sigma \times \Sigma$: une relation binaire sur Σ, la relation de transition.*

Pour deux états σ_1, σ_2, $(\sigma_1, \sigma_2) \in \rightarrow$ est noté $\sigma_1 \rightarrow \sigma_2$ et indique l'existence d'une transition de l'état σ_1 à l'état σ_2.

Définition . *Système de transitions étiqueté. Un système de transitions étiqueté est défini par un quadruplet* $(\Sigma, \Sigma_0, \mathcal{L}, \rightarrow)$ *où :*
- Σ *: un ensemble d'états ;*
- Σ_0 *: l'ensemble des états initiaux ;*
- $\mathcal{L}$ *: un alphabet, un ensemble d'étiquettes (de symboles) ;*
- $\rightarrow \subseteq \Sigma \times \mathcal{L} \times \Sigma$ *: la relation de transition étiquetée.*

$\sigma_1 \xrightarrow{\alpha} \sigma_2$ *définit une transition entre* σ_1 *et* σ_2 *étiquetée par* α*.*

Une étiquette représente généralement l'entrée attendue pour passer de l'état σ_1 à σ_2, une condition qui doit être vraie ou le nom de l'action prise durant la transition.

Un système de transitions est caractérisé par deux propriétés :
- fini ou infini : selon que le nombre d'états est fini ;
- déterministe si la relation de transition (en considérant ou pas les étiquettes) est une fonction, indéterministe sinon.

Définition . *Exécution d'un système de transitions. Une exécution* σ *d'un système* $S = (\Sigma, \Sigma_0, \rightarrow)$ *est définie par une séquence d'état* $\{\sigma_i | i \in \mathbb{N}\}$ *telle que :*

$$\sigma_0 \in \Sigma_0 \ \wedge \ \forall i \in \mathbb{N} : \sigma_i \rightarrow \sigma_{i+1}$$

σ_0 *est l'état initial de cette exécution. On note* $[\![S]\!]$ *l'ensemble des exécutions définies par le système* S*.*

$[\![S]\!]$ est donc la sémantique d'un système S. On note $\nu.\sigma$ la séquence des valeurs prises par ν dans l'exécution σ. Finalement, pour une propriété ϕ, on note $\sigma \vDash \phi$ si l'exécution σ vérifie ϕ.

PROPRIÉTÉS D'UN SYSTÈME DE TRANSITIONS On veut donner des propriétés et notamment des propriétés temps réel d'un système. On s'intéresse aux propriétés définies dans le cadre du modèle de trace et donc sur des séquences d'états. On définit quand une spécification vérifie une propriété et on donne les différentes propriétés qu'une spécification peut être amenée à vérifier. Une propriété est elle-même une spécification et a donc une sémantique. Une propriété est donc défini par un ensemble d'exécutions. On définit tout d'abord la vérification d'une propriété par une exécution et une spécification.

Définition . *Une exécution* σ *vérifie une propriété* ϕ *si l'on a :*

$$\sigma \vDash \phi \triangleq \sigma \in [\![\phi]\!]$$

Définition . *Une spécification* S *vérifie la propriété* ϕ *si on a :*

$$S \vDash \phi \triangleq [\![S]\!] \subseteq [\![\phi]\!]$$

Pour s'assurer qu'une spécification vérifie une propriété, on compare leurs sémantiques opérationnelles. Lors du cycle de développement, on compare par exemple deux spécifications quand on veut reformuler la spécification originale dans un autre formalisme. Afin de contrôler cette étape, on vérifie alors que la reformulation de la spécification vérifie toujours la spécification originale.

Pour une spécification, une autre propriété est sa satisfiabilité :

Définition . *Satisfiabilité. Une spécification* S *est satisfiable si :*

$$[\![S]\!] \neq \varnothing$$

Une spécification est satisfiable s'il existe une exécution la vérifiant. Si elle n'est pas satisfiable alors il est inutile de chercher à construire une implantation de ce système.

Afin de décrire plus précisément le comportement d'un système, il existe deux grandes classes de propriétés : les propriétés de vivacité et les propriétés de sûreté, classes dont une définition est donnée entre autre par [AS85]. Ces définitions sont données en se référant aux traces d'un système.

Définition . *Sûreté. Une propriété* P *est une propriété de sûreté si pour toute trace ne vérifiant pas* P*, il existe un préfixe fini tel qu'aucune trace prolongeant ce préfixe vérifie* P*.*

C'est donc une propriété qui s'invalide sur un préfixe fini. Elle traduit le besoin de ne jamais aller dans un état d'erreur, c'est-à-dire qu'un état erroné ne se produira pas. La vérification d'une propriété de sûreté se fait par analyse de chaque état du système.

Définition . *Vivacité. Une propriété* P *est une propriété de vivacité si toute trace finie peut être prolongée en une trace vérifiant* P*.*

Pour une propriété de vivacité, on traduit le besoin de passer par certains états. Un état attendu va finir par arriver. La vérification d'une propriété de vivacité peut alors se faire par le parcours de l'ensemble des exécutions définies par le système.

Proposition . *Toute propriété décrite par un ensemble de traces d'exécution est égale à la conjonction d'une propriété de sûreté et d'une propriété de vivacité.*

Cette propriété énoncée par [AS85] indique donc que les propriétés de sûreté et de vivacité suffisent à décrire l'ensemble des propriétés. On peut cependant définir d'autres sous ensembles de propriétés qui sont des cas particuliers de propriétés de sûreté ou de vivacité. Par exemple, dans le cadre d'un système concurrent, on s'assure de l'équité du système. C'est-à-dire que lorsqu'un processus demande l'accès à une ressource, il finit par l'obtenir. Il s'agit d'une propriété de vivacité. On parle aussi de propriétés d'atteignabilité. Il s'agit de pouvoir s'assurer que, depuis un état, un autre état peut être atteint.

PROPRIÉTÉS TEMPS RÉEL. Les propriétés temps réel peuvent être exprimées comme des échéances sur une opération. On cherche alors à s'assurer qu'un événement va se produire dans un temps donné. La réponse du système doit avoir lieu dans le temps imparti. A première vue on peut penser que de telles propriétés sont des propriétés de vivacité. Pour vérifier ses propriétés, il suffit de vérifier les états du système dans le temps imparti et donc sur un préfixe fini pour vérifier la présence de l'état ou l'occurrence de l'événement attendu. On a donc une propriété de sûreté car on parcourt un futur fini. Ceci est vrai uniquement si on est assuré que chaque unité de temps passe en un nombre fini d'états. L'échéance finit alors forcément par arriver. Ceci est une hypothèse de vivacité du système dite hypothèse non Zénon. Sous cette hypothèse les propriétés temps réel classiques basées sur une mesure du temps sont des propriétés de sûreté.

COMPARAISON DE SYSTÈMES DE TRANSITIONS. Pour étudier les systèmes de transition, on se base notamment sur les principes de simulation et de bisimulation

qui permettent de comparer le comportement de deux systèmes et donc de deux spécifications. Informellement, un système S_1 simule un système S_2 si S_1 autorise tous les comportements autorisés par S_2. Deux systèmes sont bisimilaires s'ils définissent exactement les mêmes comportements. La définition classique que l'on donne concerne un seul système de transitions, comparant les comportements possibles depuis deux états. Cette définition est facilement étendue à la comparaison de deux systèmes en étudiant l'union de ces systèmes.

Définition . *Relation de simulation. Pour un système de transitions* $(\Sigma, \Sigma_0, \mathcal{L} \rightarrow)$*, une relation* R *entre états est une relation de simulation si pour tout couple d'états* $(\sigma_1, \sigma_2) \in R$*, pour tout* $\alpha \in \mathcal{L}$*, et pour tout* $\sigma_1' \in \Sigma$ *tels que :*

$$\sigma_1 \xrightarrow{\alpha} \sigma_1'$$

alors il existe $\sigma_2' \in \Sigma$ *tel que*

$$\sigma_2 \xrightarrow{\alpha} \sigma_2'$$

et $(\sigma_1', \sigma_2') \in R$*. Alors* σ_2 *simule* σ_1*.*

Définition . *Relation de bisimulation. Pour un système de transitions* $(\Sigma, \Sigma_0, \mathcal{L}, \rightarrow)$*,* R *est une relation de bisimulation si* R *et* R^{-1} *sont des relations de simulations. C'est-à-dire que pour tout couple d'état* $(\sigma_1, \sigma_2) \in R$ *et pour tout* $\alpha \in \mathcal{L}$ *on a :*

- *pour tout* $\sigma_1' \in \Sigma$ *tel que :*
  $\sigma_1 \xrightarrow{\alpha} \sigma_1'$
  alors il existe $\sigma_2' \in \Sigma$ *tel que*
  $\sigma_2 \xrightarrow{\alpha} \sigma_2'$ *et* $(\sigma_1', \sigma_2') \in R$*;*
- *pour tout* $\sigma_2' \in \Sigma$ *tel que :*
  $\sigma_2 \xrightarrow{\alpha} \sigma_2'$
  alors il existe $\sigma_1' \in \Sigma$ *tel que*
  $\sigma_1 \xrightarrow{\alpha} \sigma_1'$ *et* $(\sigma_1', \sigma_2') \in R$*.*

LE TEMPS DANS LES SYSTÈMES DE TRANSITIONS. Pour l'étude des exécutions définies par un système temps réel, on s'intéresse à la trace temporisée du système.

Définition . *Trace temporisée. Soit un système de transitions* $S = (\Sigma, \Sigma_0, \rightarrow)$ *et un ensemble* $\mathbb{T}$ *muni d'une relation d'ordre total* $\leq$*. Une trace temporisée est une séquence de couples* (σ_i, t_i) *de* $\Sigma \times \mathbb{T}$ *telle que :*

$$\forall i \in \mathbb{N} : \sigma_i \rightarrow \sigma_{i+1} \ \wedge \ t_i \leq t_{i+1}$$

$\mathbb{T}$ est l'espace des temps et est généralement $\mathbb{N}$ ou $\mathbb{R}^+$. L'évolution des valeurs des t_i donne l'évolution du temps lors de l'exécution. Cette évolution dépend du modèle de temps défini par le système de transitions.

2.5.2 *Algèbres de processus*

Les algèbres de processus sont une famille de langages formels. Ils sont généralement utilisés afin de modéliser les systèmes concurrents ou distribués. Ces langages définissent un ensemble d'opérateurs pour décrire le choix entre plusieurs processus, la

composition parallèle de plusieurs processus ou encore la synchronisation par communication entre processus. L'interprétation du système donne un système de transitions.

Les algèbres de processus les plus connues sont CSP (Communicating Sequential Processes) [Hoa78] par Hoare, CCS (Calculus of Communicating Systems) [Mil82] par Milner ou encore LOTOS (Language Of Temporal Ordering Specification) [ED89].

EXTENSIONS TEMPORISÉES. Ces algèbres de processus ont été étendues avec l'introduction d'un modèle de temps et des traces temporisées. Les techniques de validation des systèmes spécifiés sont adaptées pour prendre en compte l'utilisation de traces temporisées. CCS est étendu pour donner Timed CCS [Yi91] en introduisant un unique opérateur, un préfixe de délai. Ce préfixe paramétré par une durée associe le passage d'un état à un autre à l'écoulement de cette durée.

CSP est étendue pour donner Timed CSP [RR88]. Les transitions sont étiquetées par des délais pour une sémantique semblable au préfixe de délai de Timed CCS. De plus certains opérateurs sont étendus et de nouveaux opérateurs sont introduits, par exemple un opérateur de time-out pour mettre en attente une action durant un temps borné ou un opérateur *WAIT* permettant de faire s'écouler le temps. Dans RT Lotos [CdA95], plusieurs opérateurs sont aussi introduits permettant de retarder une action d'un délai déterministe ou indéterministe mais aussi permettant de forcer l'exécution d'une action dans un intervalle. Des solutions pour la composition d'intervalles sont données. Dans ces algèbres de processus temporisées un temps implicite est utilisé. On ne peut directement manipuler la valeur de la date courante.

2.5.3 *Automates Temporisés*

Un automate est une machine à état finie. Il peut être représenté par un graphe. Une définition formelle d'un automate est la suivante :

Définition . *Automate. Un automate est défini par un quintuplet* $A = (N, \mathcal{L}, \rightarrow, I, Q)$ *avec :*
- N *: ensemble fini d'états ;*
- $\mathcal{L}$ *: un alphabet, un ensemble fini de symboles ;*
- $\rightarrow$: $(N \times \mathcal{L}) \times N$ *: la relation de transition ;*
- I *: l'ensemble des états initiaux ;*
- Q *: l'ensemble des états accepteurs.*

Dans la représentation par un graphe, chaque nœud est un état et les arcs définissent la relation de transition du système. Un automate est donc un système de transitions étiqueté. A partir de l'état initial, on utilise un mot c'est-à-dire une séquence finie de symboles de l'alphabet. Ce mot détermine avec la relation de transition l'enchaînement des états pris par le système, les symboles du mot étant lus les uns après les autres. Le mot est reconnu si tout au long de la lecture du mot il existe une transition correspondant au symbole lu. De plus, l'état final du système doit faire partie des états accepteurs. L'automate donné figure 1 reconnaît les mots constitués de a et de b et finissant par un b.

Il existe plusieurs types d'automates. On cite les automates de Büchi [Tho90] qui sont une extension à des mots infinis. Pour cela, la sémantique des états accepteurs est modifiée, un mot est accepté si l'exécution passe infiniment souvent dans les états accepteurs. L'ensemble de ces états étant fini, c'est équivalent au passage infiniment souvent par l'un des états accepteurs .

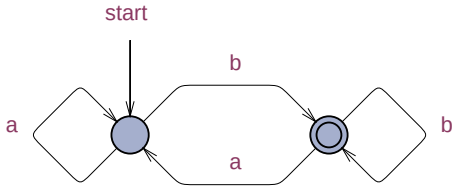

FIG. 1: Un exemple d'automate

Les automates temporisés sont le type d'automate qui nous intéresse le plus. On introduit la forme la plus courante définie par [AD94]. Le temps est introduit par l'utilisation d'un ensemble d'horloges qui sont des variables positives, prenant leurs valeurs dans $\mathbb{R}^+$.

Définition 1. *Automate temporisé. Un automate temporisé est défini par un sextuplet* $A = (N, \mathcal{L}, C, T, I, Q)$ *où :*
- N *: un ensemble fini d'états ;*
- $\mathcal{L}$ *: un alphabet, un ensemble fini de symboles ;*
- C *: un ensemble fini d'horloges ;*
- I *: l'ensemble des états initiaux ;*
- $T \subset N \times N \times \mathcal{L} \times 2^C \times \Phi(C)$ *: l'ensemble des transitions ;*
- Q *: l'ensemble des états accepteurs.*

L'ensemble 2^C *définit les horloges remises à zéro lors de la prise de la transition. Pour un ensemble d'horloges* C, $\Phi(C)$ *définit l'ensemble des conditions* δ *:*

$$\delta := z \sim c \mid \delta_1 \wedge \delta_2$$

où $c \in \mathbb{Q}$ *est un rationnel,* $z \in C, \sim \in \{<, \leq, =, \neq, \geq, >\}$. *Pour une condition* δ *et un vecteur de valeurs* u *prises par les horloges de* C, *on note* $\delta(u)$ *le prédicat indiquant que les valeurs de* u *vérifient* δ.

Définition 2. *Système de transitions défini par un automate temporisé. Un automate temporisé définit un système de transitions où un état du système est défini par un couple* $\langle n, u \rangle$ *donnant l'état courant* n *et les valeurs* u *des horloges. On note* $u' = [r]u + c$ *l'affectation d'horloge qui ajoute* c *aux valeurs* u *prises par les horloges de* C, *excepté les horloges de* r *qui sont remises à zéro. La relation de transition* $\rightarrow$ *d'un automate temporisé est définie par* $\langle n, u \rangle \rightarrow \langle n', u' \rangle$ *si :*

$$\exists r \subseteq C, \exists a \in \mathcal{L}, \exists \delta \in \Phi(C) : \langle n, n', a, r, \delta \rangle \in T \text{ et } \exists d \in \mathbb{R}^+ : \delta([\varnothing]u + d) \wedge u' = [r]u + d$$

L'évolution des horloges se fait au même rythme que l'évolution du temps qui est implicite. On peut donc avoir une mesure du temps global avec une horloge jamais remise à 0. Une transition existe lorsque le passage du temps permet d'arriver dans un état où les valeurs des horloges vérifient la condition associée à une transition. L'étiquette associée à cette transition est alors lue et on change alors de nœud en remettant à jour l'ensemble des horloges spécifié par cette transition. Un automate temporisé est déterministe si les contraintes sur les horloges et les étiquettes des différentes transitions quittant un état sont mutuellement exclusives. Un exemple d'automate temporisé est donné figure 2.

D'autres modèles d'automates temporisés existent, notamment celui utilisé dans l'outil de vérification Uppaal [LPY97]. Par rapport à la définition utilisée ici, les nœuds des automates d'Uppaal peuvent avoir un invariant sur les horloges comme propriété.

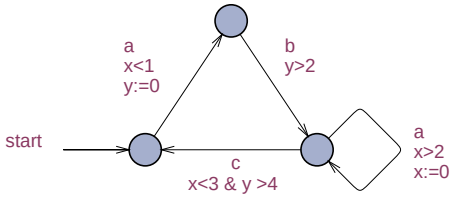

FIG. 2: Un exemple d'automate temporisé

Types de mise à jour	Sans contraintes diagonales	Avec contraintes diagonales
$x := c; x := y$	PSpace complet	PSpace complet
$x := x + 1$		Indécidable
$x := y + c$		
$x := x - 1$	Indécidable	
$x :< c$	PSpace complet	PSpace complet
$x :> c$		Indécidable
$x :\sim y + c$		
$y + c <: x :< y + d$		
$y + c <: x :< z + d$	Indécidable	

FIG. 3: Résultats de décidabilité

Cet invariant est utilisé pour forcer un changement d'état avant que l'invariant ne soit plus respecté. De plus, il est possible de mettre plusieurs automates en concurrence ou de les synchroniser par rendez vous.

DÉCIDABILITÉ. Le problème de la satisfiabilité pour les automates dont la relation de transition est défini par la définition 2 est décidable et PSpace complet. Cette vérification s'appuie sur l'utilisation de zones ou régions définies par un ensemble de contraintes sur les horloges, la prise de transition définit un nouvel ensemble de contraintes et une nouvelle zone. Pour du model checking, on construit l'intégralité de ces régions qui est fini mais exponentiel par rapport au nombre d'horloges. On obtient cependant un graphe abstrait de l'ensemble des états atteignables dans un automate temporisé. Cette méthode est détaillée dans [BY04]. L'outil de model checking Uppaal utilise cette méthode pour vérifier des automates temporisés. Les propriétés qui sont vérifiées sont spécifiées dans une logique temporelle proche de CTL .

D'autres types d'automates temporisés existent, différenciés par la définition de la relation de transition et notamment par les mises à jour des horloges et le type de conditions sur ces horloges. On peut utiliser des contraintes diagonales, c'est-à-dire évaluant la différence $x - y \sim c$ entre deux horloges x et y par rapport à c et où $\sim \in \{<, \leq, =, \neq, \geq, >\}$. Il est aussi possible de mettre à jour une horloge avec n'importe quelle valeur de manière déterministe, par exemple $z := x + c$ affecte la valeur de l'horloge x plus une constante à l'horloge z, ou indéterministe, par exemple $z :< c$ affecte une valeur entre 0 et c à l'horloge z. La figure 3 reprend le tableau donné dans [FPY02] et résume les résultats de décidabilité pour la satisfiabilité de l'automate. Dans chaque cas, il s'agit de construire une partie du graphe des régions abstrayant les états atteignables. Les différentes mises à jour données sont ajoutées à la remise à zéro.

EXPRESSIVITÉ. L'expressivité des différents modèles est aussi donnée par [FPY02]. Le modèle le moins expressif est celui d'un automate classique défini par la définition 1 et dont la relation de transition est déterministe. On regroupe ensuite les modèles décidables ayant des mises à jour des horloges déterministes. Ils sont équivalents aux automates temporisés classiques définis par [AD94]. En terme d'expressivité, on trouve ensuite les automates ayant des mises à jour indéterministes mais libres de contraintes diagonales puis ceux avec ce type de contraintes. Pour finir, on a les classes indécidables.

Finalement, on utilise des automates temporisées pour réaliser des analyses d'ordonnancement [FPY02]. Cette technique est implantée dans l'outil Times. On peut notamment modéliser le déclenchement d'une tâche selon l'état d'un automate.

2.5.4 *Réseaux de Petri temporisés*

Un réseau de Petri est constitué de places, de transitions et d'arcs orientés. Les arcs relient une transition à une place, ou inversement une place à une transition. Chaque place contient un nombre positif ou nul de jetons. La disposition des jetons dans les différentes places définit le marquage du système autrement dit l'état courant du système. Une transition est définie par un passage de jetons d'une place à une autre le long des arcs. Si chaque place d'entrée des arcs entrants dans une transition possède suffisamment de jetons, cette transition peut être tirée. Alors toutes les places reliées par arc sortant de cette transition reçoivent des jetons supplémentaires.

Définition 3. *Réseau de Petri. Un réseau de Petri est défini par un quintuplet* (S, T, F, W, M_0) *:*
- S *: l'ensemble des places ;*
- T *: l'ensemble des transitions ;*
- $F \subset (S \times T) \cup (T \times S)$ *: l'ensemble des arcs ;*
- $W : F \to \mathbb{N}$ *: la valuation de chaque arc. Cette fonction associe à chaque arc de* F *un entier positif. Ils indiquent combien de jetons seront consommés pour activer la transition en sortie de cet arc ou combien de jetons seront produits si la transition en entrée est tirée.*
- $M_0 : S \to \mathbb{N}$ *est le marquage initial c'est-à-dire le nombre de jetons dans chaque place à l'état initial*

Définition 4. *Système de transitions défini par un réseau de Petri. Le marquage du système définit l'état courant du système, la relation de transition* $\to$ *est une relation entre deux marquages. Une transition* t *est tirée et définit une transition entre deux marquages* M *et* M' *si on a :*

$$\forall s : (s,t) \in F, (t,s) \notin F : M'(s) = M(s) - W((s,t)) \wedge$$
$$\forall s : (t,s) \in F, (s,t) \notin F : M'(s) = M(s) + W((t,s)) \wedge$$
$$\forall s : (s,t) \in F, (t,s) \in F : M'(s) = M(s) - W((s,t)) + W((t,s)) \text{ et à condition que l'on ait :}$$
$$\forall (s,t) \in F : M(s) \geq W((s,t))$$

Une transition t peut être tirée si et seulement si elle est sensibilisée par les jetons en entrée.

Un exemple de réseau de Petri et un tir de transition sont donnés figure 4. En sémantique faible, contrairement à la sémantique forte, le tir d'une transition autorisée n'est pas automatique. De plus si deux transitions sont tirables et indépendantes, elles peuvent être tirées séquentiellement ou en parallèle.

On s'intéresse aux propriétés suivantes des réseaux de Petri :
- la causalité et le parallélisme ;
- l'attente de jetons, la bonne circulation de ces jetons et les situations de blocages ;

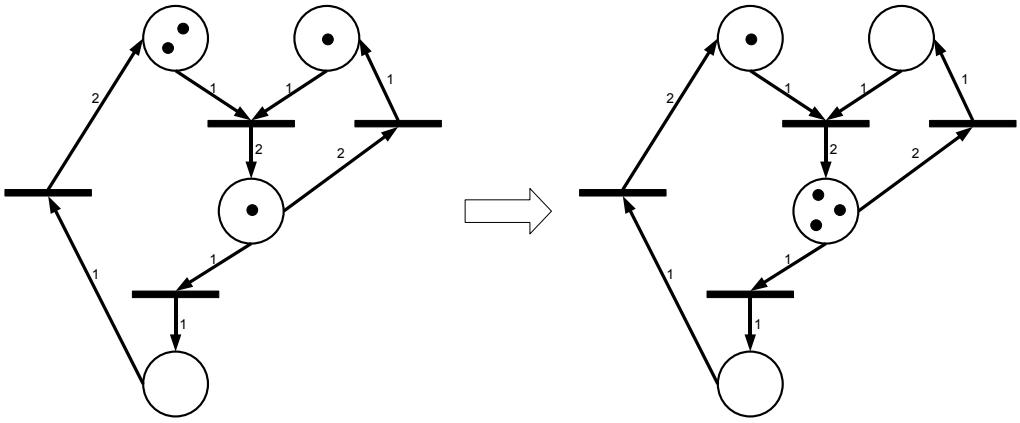

FIG. 4: Un exemple de réseau de Petri

- l'évolution du nombre de jetons et ainsi l'évolution du nombre de ressources et du parallélisme ;
- le non-déterminisme.

La vérification des réseaux de Petri se concentre donc sur la recherche de bornes sur le nombre de jetons, la vivacité des différentes transitions et la réinitiabilité du réseau. Ces propriétés sont décidables. L'outil Tina [BLRV07] permet leur vérification en construisant un espace abstrait des états atteignables. Cette espace peut être utilisé pour du model checking et la vérification des propriétés des exécutions définies par un réseau de Petri.

EXTENSIONS TEMPORISÉES. Plusieurs extensions des réseaux de Petri existent, notamment certaines permettent de modéliser le temps et des propriétés temps réel. Ces extensions sont regroupées en deux familles, les réseaux de Petri temporisés introduits par [Ram74] et les réseaux de Petri temporels introduits par [Mer74]. Pour les réseaux de Petri temporisés, le temps est utilisé pour donner une durée minimale de tir pour une transition ou un temps de séjour minimal pour un jeton dans une place. Ces réseaux de Petri sont une sous-classe des réseaux de Petri temporels.

Les premiers réseaux de Petri temporels associent à chaque transition un intervalle de temps, ce sont les réseaux T-temporels. Une transition ne peut être tirée que si elle a été sensibilisée en continu pendant la durée minimale de l'intervalle. Elle doit être tirée avant d'atteindre le maximum sauf si le tir d'une autre transition entraîne l'arrêt de sa sensibilisation. Les réseaux de Petri T-temporels sont utilisés pour l'étude des protocoles de communication. Concernant la décidabilité, les bornes sur le nombre de jetons sont décidables. Les propriétés d'accessibilité et la vérification de propriétés exprimées dans la logique RTCTL sont décidables uniquement dans le cas où le réseau de Petri est borné. Cependant, pour ces réseaux de Petri, le temps est associé uniquement à la sensibilisation d'une transition. Un jeton peut rester indéfiniment dans une place en attente d'une synchronisation et sans qu'il y ait d'information sur le temps d'attente avant synchronisation. L'outil Tina permet l'analyse des propriétés des réseaux de Petri T-temporels.

Pour palier cela, les réseaux de Petri P-temporels associent un intervalle de temps aux places. Chaque jeton ne peut quitter la place qu'après avoir passé une durée minimale

de temps dans cette place et il doit la quitter avant le maximum de l'intervalle de la place. S'il ne peut pas la quitter car aucune transition n'est sensibilisée, le jeton est déclaré "mort". Une horloge est donc associée à chaque jeton. Le but est de vérifier l'absence d'exécutions créant des jetons morts ce qui traduit un manque de synchronisation. L'expressivité des deux types de réseaux de Petri temporels n'est pas comparable [BR08].

2.5.5 *Le langage TLA (Temporal Logic of Actions)*

Le langage TLA (Temporal Logic of Actions) est un langage développé par Leslie Lamport [Lam94] permettant de spécifier des systèmes de transition. Ce langage permet de plus de spécifier des propriétés d'équité. En TLA, les programmes sont représentés par des formules de logique temporelle et plus précisément par une formule de la forme :

$$\mathtt{Init} \wedge \Box[A]_t \wedge \mathtt{Equite}$$

$\mathtt{Init}$ est un prédicat sur l'état initial du système. Par exemple $\mathtt{Init} \triangleq (x = 0)$ indique que la variable x est initialisée à 0.

Une action A est un prédicat entre deux états du système liés par la relation de transition. Ce prédicat est défini par des variables primées qui donnent l'état des variables dans l'état suivant et les variables non primées qui donnent l'état des variables dans l'état courant. Par exemple l'action $x' = x + 1$ est vérifiée pour deux états tel que le second a x incrémenté de 1 par rapport au premier. On construit une action par disjonction et conjonction d'autres actions.

Les variables t en indice de l'action A sont des variables impliquées dans la définition de l'action A. Cet indice autorise le bégaiement, c'est-à-dire que toute transition qui ne modifie pas ces variables est autorisée permettant l'entrelacement. Ainsi $[x' = x + 1]_x$ autorise une transition où x n'est pas modifié mais où d'autres variables le sont. Finalement, le symbole $\Box$ indique que toute transition du système doit respecter $[A]_t$.

$\mathtt{Equite}$ spécifie les propriétés d'équité, répondant au besoin de spécifier que certaines actions finissent par arriver. On utilise principalement l'équité faible $WF_t(A)$ et l'équité forte $SF_t(A)$. $WF_t(A)$ est une propriété d'équité faible, c'est-à-dire que lors de l'exécution du système l'action $A \wedge t' \neq t$ ne peut pas être faisable en continu sans que la transition associée finisse par être prise. La vivacité forte peut aussi être utilisée. $SF_t(A)$ indique que $A \wedge t' \neq t$ ne peut pas être faisable infiniment souvent sans que la transition associée ne soit prise.

On utilise le langage TLA pour vérifier qu'une spécification est bien définie, c'est-à-dire qu'elle définit un ensemble non vide d'exécutions, ou alors qu'un système implante une spécification exprimée dans la logique temporelle LTL. TLC (TLA+ Model Checker) est un outil permettant de vérifier des spécifications TLA par model checking.

Le temps dans TLA est introduit via une variable now dont l'écoulement est à décrire dans les actions décrivant le système. On manipule donc directement la variable temps et non pas par l'intermédiaire d'horloges. Ce principe simple est expliqué dans [AL94].

2.6 CONCLUSION

La définition classique d'un système temps réel repose sur la correction temporelle des opérations de ce système. Dans une spécification, les propriétés temps réel caractérisent

donc les temps de calcul ou de communication. Dans notre approche, on définit un système temps réel de la manière suivante :

> La validité d'un système temps réel dépend de la validité temporelle de ses données.

Les propriétés temps réel sont alors exprimées comme des conditions de validité temporelle des valeurs prises par les variables du système. Ces conditions de validité dépendent alors de la sémantique de cette variable et définissent par exemple le rythme de mise à jour des variables. De plus, dans le cas d'un système où les valeurs de certaines variables sont utilisées pour calculer les valeurs d'autres résultats, on veut définir la validité temporelle des valeurs résultats en fonction de la validité temporelle des valeurs utilisées en entrée pour le calcul.

Les approches usuelles pour les systèmes temps réel reposent évidemment sur la définition classique du temps réel et définissent les propriétés temps réel comme des propriétés sur les caractéristiques temporelles des opérations et pas sur les données. Cette différence ne nous permet pas de réutiliser directement ces approches pour l'expression des propriétés. On veut de plus procéder à une vérification formelle de la spécification d'un système et le cadre des approches par les modèles ne nous permet pas de définir formellement la sémantique de nos propriétés. Pour ces raisons, la définition des propriétés et de leur sémantique est donnée par la suite dans un cadre non usuel pour l'étude des systèmes temps réel.

On s'intéresse alors à la relation d'observation formalisée par [Cha97]. Cette relation permet de modéliser les communications dans les systèmes distribués mais sans donner de propriétés temps réel sur cette communication. On utilise aussi cette relation pour pouvoir modéliser les calculs effectués dans un système. Un ensemble de relations d'observation définit des chemins le long desquels les valeurs d'une variable sont propagées pour définir les valeurs d'autres variables. Ces chemins donnent donc des dépendances entre les valeurs des différentes variables du système. On définit les propriétés temps réel du système sur ces chemins de propagation introduits par les relations d'observation. Il s'agit de limiter le décalage temporel introduit par cette propagation et de caractériser les dépendances entre la validité temporelle d'une variable et la validité temporelle des autre variables du système.

Les systèmes de transitions permettent de donner la sémantique opérationnelle d'une spécification en construisant explicitement les exécutions du système. Sous certaines conditions, il est alors possible de vérifier les propriétés d'une spécification par exploration des états des exécutions. Les systèmes de transitions tels que les automates ou les réseaux de Petri, bien qu'utilisés pour l'étude de propriétés temps réel, ne permettent pas directement de décrire la propagation des valeurs des variables dans le système et de spécifier les propriétés de ces valeurs. On n'utilise pas ces types de système de transitions pour définir les propriétés d'un système mais on utilise cependant les systèmes de transitions pour procéder à la vérification d'une spécification.

3

OUTILS POUR LA SPÉCIFICATION D'UN SYSTÈME

On donne dans ce chapitre les définitions des concepts utilisés par la suite pour exprimer la spécification d'un système. On définit les relations utilisées pour définir les liens entre les variables du système et les moyens de caractériser le comportement d'une variable lors d'une exécution. On commence par définir la représentation du temps dans le système et introduire les horloges qui sont utilisées par la suite pour définir le décalage logique introduit lors de la propagation des valeurs des variables. On donne ensuite la définition de la relation d'observation et comment on utilise cette relation pour modéliser les communications et les calculs et décrire l'architecture d'un système. On introduit le profil temporel et les occurrences d'une variable permettant de décrire le comportement d'une variable le long d'une exécution. Finalement, on construit à partir d'un ensemble de relation d'observation les chemins de propagation des valeur des variables dans un système.

3.1 DÉFINITIONS PRÉLIMINAIRES

On se place dans le cadre du modèle de traces qui est introduit pour l'étude des systèmes de transitions section 2.5.1, page 28. On définit alors un système comme une spécification ayant une sémantique associée permettant de donner l'ensemble des exécution définies par la spécification. Les propriétés du systèmes sont décrites sur les exécutions du système définies comme des séquences d'états. On considère ici uniquement des exécutions infinies et donc des séquences $\sigma_0, \sigma_1, \sigma_2, \ldots \sigma_i, \sigma_{i+1}, \ldots$ d'états indexés par $\mathbb{N}$.

3.1.1 *Introduction du temps*

Afin d'intégrer le temps dans les spécifications de système que nous allons donner, on introduit une référence explicite au temps. On utilise comme dans [AL94] une variable T croissante et non bornée, prenant valeur dans un ensemble infini muni d'une relation d'ordre total. L'ensemble des valeurs de T choisi est $\mathbb{N}$ et donc discret.

Définition 5. *Variable temps. L'évolution du temps est définie par une variable* T *telle que pour toute exécution* σ *:*

- T *prend valeur dans* $\mathbb{N}$ *:*
  $\forall i \in \mathbb{N} : T.\sigma_i \in \mathbb{N}$
- T *est croissante :*
  $\forall i \in \mathbb{N} : T.\sigma_i \leq T.\sigma_{i+1}$
- T *est initialisée à* 0 *:*
  $T.\sigma_0 = 0$

La variable T permet une spécification explicite du temps qui passe. On n'utilise pas les traces temporisées mais on utilise la valeur de T dans chaque état pour pouvoir spécifier et étudier le comportement temporel du système. Le choix d'un temps discret permet par la suite de faciliter l'analyse d'une spécification. De manière générale, ce choix d'un temps discret par rapport à un temps continu est discuté dans [LSV96].

Cependant on étudie les exécutions du système comme une séquence d'états, ce qui discrétise les valeurs de T auxquelles on s'intéresse.

On définit les systèmes non Zénons qui assure une évolution du temps non bornée.

Définition 6. *Système non Zénon. Un système S est dit non Zénon si :*

$$\forall \sigma \in [\![S]\!], \forall n \in \mathbb{N}, \exists i \in \mathbb{N} : T.\sigma_i \geq n$$

L'unité de temps n'est pas définie à l'avance. Cependant, on ajoute une propriété précisant que l'échelle de temps que l'on utilise est suffisamment précise pour observer tous les états du système. Cela signifie que chaque pas de calcul et donc modification de l'état du système implique l'écoulement du temps et donc une évolution de T. On appelle cette propriété la propriété de séparabilité.

Définition 7. *Séparabilité. Une exécution σ est dite séparable si pour toute variable v on a :*

$$\forall i, j : T.\sigma_i = T.\sigma_j \Rightarrow v.\sigma_i = v.\sigma_j$$

Un système est dit séparable si toutes les exécutions de sa sémantique sont séparables.

On considère uniquement des exécutions séparables ce qui permet de dater chaque transition du système. On s'intéresse de plus à des variables dont les valeurs évoluent au cours d'une exécution. Ce qui conduit à définir la vivacité d'une variable :

Définition 8. *Variable vivace. Une variable v d'un système est dite vivace si pour toute exécution σ de ce système on a :*

$$\forall i \in \mathbb{N}, \exists j > i : v.\sigma_j \neq v.\sigma_i$$

Si une variable est vivace et que les exécutions d'un système sont séparables alors la variable T est vivace. On déduit une propriété qui concerne l'hypothèse d'un système non Zénon.

Proposition 1. *Soit S un système séparable et une variable v de S vivace, alors S est non Zénon.*

Démonstration. Soit S un système séparable et v une variable vivace. On considère une exécution σ de S. La variable T prend valeur dans $\mathbb{N}$, on utilise alors un raisonnement par récurrence. L'hypothèse de récurrence est donc pour un entier n :

$$\exists i : T.\sigma_i \geq n$$

Étant donné qu'à l'état initial $T.\sigma_0 = 0$, l'hypothèse est vérifiée au rang 0. Supposons qu'elle soit vérifiée au rang n. Alors $\exists i : T.\sigma_i = n$. Comme v est vivace on a :

$$\begin{array}{ll} & \exists j > i : v.\sigma_j \neq \sigma_i \\ \Rightarrow & \quad \{S \text{ est séparable et par contraposée}\} \\ & \exists j > i : T.\sigma_j \neq T.\sigma_i \\ \Rightarrow & \quad \{T \text{ est croissante}\} \\ & \exists j > i : T.\sigma_j > T.\sigma_i \\ \Rightarrow & \quad \{\text{définition de } i\} \\ & \exists j > i : T.\sigma_j > n \\ \Leftrightarrow & \quad \{\text{propriété de } \mathbb{N}\} \\ & \exists j > i : T.\sigma_j \geq n + 1 \end{array}$$

L'hypothèse de rang $(n + 1)$ est donc vérifiée et donc par récurrence étendue à tout rang. □

Même si nous travaillons avec les valeur de T dans $\mathbb{N}$, une exécution qui ne prend pas toutes les valeurs de $\mathbb{N}$ est possible. On pourrait définir une exécution où dans chaque état les valeurs de T sont paires. Un parcours de ces états ne permet pas de vérifier les propriétés du système dans les instants impairs. On veut donc que toutes les valeurs de $\mathbb{N}$ soient prises lors de l'exécution afin d'étudier l'état du système à tout instant. On suppose donc par la suite que l'on a la propriété suivante :

Axiome . *Pour toute exécution, la variable* T *vérifie la propriété suivante :*

$$T.\sigma = \mathbb{N}$$

3.1.2 *Les horloges*

Afin de décrire le comportement temporel des variables du système on définit l'ensemble des horloges. Il s'agit de fonctions définies sur un ensemble muni d'un ordre total tel que $\mathbb{N}$ ou $\mathbb{R}^+$ et définissant une sous-séquence de valeurs de cet ensemble obéissant à des propriétés de sûreté et de vivacité. Considérant les fonctions que l'on définit, on note $[\mathcal{X} \to \mathcal{Y}]$ l'ensemble des fonctions de l'ensemble de départ $\mathcal{X}$ vers l'ensemble d'arrivée $\mathcal{Y}$.

Définition 9. *Horloge. Soit un ensemble $\mathcal{D}$ muni d'un ordre total $\leq$. Une horloge* c *est une fonction de $[\mathcal{D} \to \mathcal{D}]$ telle que :*

- *elle ne dépasse pas son argument :*
  $$\forall t \in \mathcal{D} : c(t) \leq t$$
- *elle est monotone et croissante :*
  $$\forall t_1, t_2 \in \mathcal{D} : t_1 < t_2 \Rightarrow c(t_1) \leq c(t_2)$$
- *elle est vivace :*
  $$\forall t_1 \in \mathcal{D} : \exists t_2 \in \mathcal{D} : c(t_2) \neq c(t_1)$$

On note $\mathcal{C}(\mathcal{D})$ l'ensemble des horloges définies sur $\mathcal{D}$

On utilise des horloges définies sur $\mathbb{N}$ mais avec cependant deux significations différentes. Dans un cas, $\mathbb{N}$ représente le domaine de définition de la variable de temps T. Les horloges caractérisent alors les décalages temporels introduits par l'architecture du système pour la propagation des valeurs prises par les différentes variables. Par ailleurs, $\mathbb{N}$ définit aussi les indices des états. Les horloges basées sur $\mathbb{N}$ représentent alors la précédence logique.

Proposition 2. *Valeur initiale des horloges sur $\mathbb{N}$. Pour toute horloge* c *de $\mathcal{C}(\mathbb{N})$, on a :*

$$c(0) = 0$$

Démonstration. Une horloge ne dépasse pas son argument donc pour une horloge c de $\mathcal{C}(\mathbb{N})$, on a $c(0) \leq 0$. Comme cette horloge prend ses valeurs dans $\mathbb{N}$, on a de plus $c(0) \geq 0$. On en déduit donc que $c(0) = 0$. □

PROPRIÉTÉS DES HORLOGES

Proposition 3. *Soit un ensemble d'horloges $\mathcal{C}(\mathbb{N})$. La composition de deux horloges de $\mathcal{C}(\mathbb{N})$ est une horloge de $\mathcal{C}(\mathbb{N})$.*

Démonstration. Soit un ensemble d'horloges $\mathcal{C}(\mathbb{N})$ et deux horloges c_1 et c_2 de cet ensemble. Alors $c_1 \circ c_2$ est bien une fonction de $[\mathbb{N} \to \mathbb{N}]$. Vérifions les trois propriétés définissant une telle fonction comme une horloge :

– soit $t_1, t_2 \in \mathbb{N} : t_1 < t_2$ alors d'après les propriétés de c_1 et c_2 on a :
$$\begin{array}{ll} & t_1 \leq t_2 \\ \Rightarrow & \{c_2 \text{ croissante}\} \\ & c_2(t1) \leq c_2(t_2) \\ \Rightarrow & \{c_1 \text{ croissante}\} \\ & c_1(c_2(t_1)) \leq c_1(c_2(t_2)) \end{array}$$
donc $c_1 \circ c_2$ est croissant ;
– soit $t \in \mathbb{N}$ alors c_2 ne dépasse pas son argument donc :
$$\begin{array}{ll} & c_2(t) \leq t \\ \Rightarrow & \{c_1 \text{ ne dépasse pas son argument}\} \\ & c_1(c_2(t)) \leq c_2(t) \leq t \end{array}$$
donc $c_1 \circ c_2$ ne dépasse pas son argument ;
– On utilise le lemme suivant, démontré à la suite, pour une horloge c de $\mathcal{C}(\mathbb{N})$:

$$\forall k \in \mathbb{N} : \exists t \in \mathbb{N} : c(t) > k$$

Soit $t_1 \in \mathbb{N}$ alors d'après le lemme et comme $c_1(c_2(t_1)) \in \mathbb{N}$, on a :
$$\begin{array}{ll} & \exists t_2 \in \mathbb{N} : c_1(t_2) > c_1(c_2(t_1)) \\ \Rightarrow & \{\text{lemme} : \exists t_3 \in \mathbb{N} : c_2(t_3) > t_2)\} \\ & \exists t_2, t_3 \in \mathbb{N} : c_2(t_3) > t_2 \ \wedge \ c_1(t_2) > c_1(c_2(t_1)) \\ \Rightarrow & \{c_1 \text{ croissante}\} \\ & \exists t_2, t_3 \in \mathbb{N} : c_1(c_2(t_3)) \geq c_1(t_2) \ \wedge \ c_1(t_2) > c_1(c_2(t_1)) \\ \Rightarrow & \{\text{simplification}\} \\ & \exists t_3 \in \mathbb{N} : c_1(c_2(t_3)) > c_1(c_2(t_1)) \end{array}$$
donc $c_1 \circ c_2$ est vivace.

On en déduit donc que $c_1 \circ c_2$ est une horloge.

On démontre le lemme par récurrence sur $\mathbb{N}$. Au rang 0, on a $c(0) = 0$. Comme c est vivace, il existe $t > 0$ tel que $c(t) \neq 0$. Comme c est croissante, on a $c(t) > 0$ et donc la propriété est vérifiée.

On suppose qu'on a un rang k tel que la propriété est vérifiée en $k - 1$. Donc il existe t tel que $c(t) > k - 1$ et comme on est dans le domaine des entiers, $c(t) \geq k$. Or, il existe $t' > t$ tel que $c(t') \neq c(t)$ et donc, comme c est croissante, $c(t') > c(t)$. Donc $c(t') > k$ et le lemme est vérifié par récurrence. □

3.2 LA RELATION D'OBSERVATION

3.2.1 *Définition*

Nous reprenons la définition de la relation d'observation telle qu'initialement donnée par [Cha97]. Cette relation a été définie pour abstraire les communications dans les systèmes distribués. Elle décrit la vision partielle et retardée qu'un site a d'un autre site dans une application répartie. Le but est de définir un outil simple pour la description et l'étude des programmes répartis. Cet outil permet de s'abstraire du protocole de communication utilisé en ne s'intéressant qu'au décalage introduit par la répartition.

Définition 10. *Relation d'observation. Soit deux expression* $'e$ *et* e*. Pour une exécution* σ*, ces deux expressions sont liées par une relation d'observation* $'e \prec e$ *si :*

$$\sigma \vDash {}'e \prec e \triangleq \exists c \in \mathcal{C}(\mathbb{N}) : \forall i \in \mathbb{N} : {}'e.\sigma_i = e.\sigma_{c(i)}$$

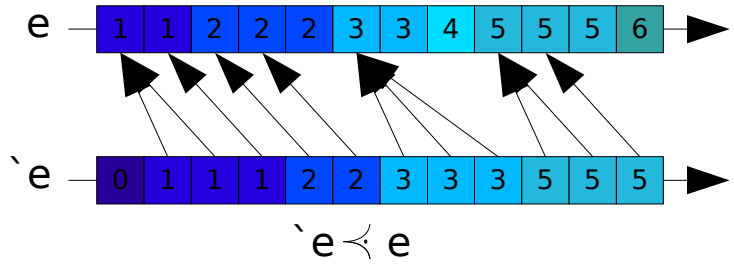

FIG. 5: Représentation d'une observation

On dit alors que l'expression ‘e est une observation de e. Au long de l'exécution et grâce aux propriétés des horloges, on exprime que les valeurs prises par l'expression ‘e sont des valeurs antérieures de e, prises dans l'ordre chronologique. De plus la vivacité des horloges implique que ‘e prend des valeurs de plus en plus récentes de e. La perte est autorisée et donc toutes les valeurs de e ne sont pas nécessairement prises par ‘e. On appelle l'expression ‘e l'*image* de l'observation et e la *source* de l'observation. On dit alors que ‘e observe e dans une exécution σ ou encore que ‘e est une observation de e. On utilise le terme *observation* dans la suite du document pour désigner une *relation d'observation*.

L'observation est illustrée figure 5. Chaque état correspond aux deux cases alignées verticalement et les valeurs prises par les expressions ‘e et e le long d'une exécution sont représentées. Les flèches représentent une possible horloge de l'observation en donnant l'état pointé par cette horloge en chaque état.

PROPRIÉTÉS On liste les propriétés de la relation d'observation, propriétés démontrées dans [Cha97].

Proposition 4. *Propriétés de la relation d'observation Soit trois expressions* e_1, e_2, e_3 *alors :*

- *dans l'état initial* σ_0 *d'une exécution, source et image d'une observation ont même valeur ;*
  $\forall \sigma : \sigma \vDash e_2 \prec e_1 \Rightarrow e_2.\sigma_0 = e_1.\sigma_0$
- *la relation d'observation est réflexive, antisymétrique et transitive ;*
  $\forall \sigma : \sigma \vDash e_1 \prec e_1$
  $\forall \sigma : \sigma \vDash e_2 \prec e_1 \ \wedge \ e_1 \prec e_2 \Rightarrow e_2.\sigma = e_1.\sigma$
  $\forall \sigma : \sigma \vDash e_3 \prec e_2 \ \wedge \ e_2 \prec e_1 \Rightarrow \sigma \vDash e_3 \prec e_1$

  c'est donc une relation d'ordre partiel.

3.2.2 *Restriction de la relation d'observation*

On veut utiliser la relation d'observation pour décrire l'architecture d'un système distribué temps réel. On s'intéresse à la propagation des valeurs de chaque variable dans le système et au retard introduit par chacune des étapes de la propagation. Ces étapes sont les communications et les calculs effectués par le système. On restreint l'usage de la relation d'observation à une définition permettant de modéliser une telle architecture.

Définition 11. *Restriction de la relation d'observation. On restreint la relation d'observation à une relation entre deux expressions définies par une variable* y*, un n-uplet de variables* $\langle x_1, \ldots, x_n \rangle$ *et une fonction de* f *tel que :*

$$\sigma \vDash y \prec f(x_1, \ldots, x_n) \triangleq$$
$$\exists c \in \mathcal{C}(\mathbb{N}) : \forall i \in \mathbb{N} : y.\sigma_i = f(x_1.\sigma_{c(i)}, x_2.\sigma_{c(i)}, \ldots, x_n.\sigma_{c(i)})$$

MODÉLISATION D'UNE COMMUNICATION PAR L'OBSERVATION Pour modéliser une communication, on définit une relation d'observation où la fonction f est la fonction identité et où on réduit le n-uplet à une seule variable. On a alors une relation d'observation $'x \prec x$ entre deux variables $'x$ et x et telle que pour une exécution σ, on a :

$$\sigma \vDash 'x \prec x \triangleq \exists c \in \mathcal{C}(\mathbb{N}) : \forall i \in \mathbb{N} : 'x.\sigma_i = x.\sigma_{c(i)}$$

Les valeurs prises par $'x$ sont une copie avec retard des valeurs prises par x. Le retard modélise le temps de communication. La perte est de plus autorisée.

MODÉLISATION D'UN CALCUL PAR L'OBSERVATION. Pour modéliser un calcul où les valeurs d'une variable y sont définies par le résultat d'une fonction f appliquée à un n-uplet de variables $\langle x_1, \ldots, x_n \rangle$, on définit une relation d'observation $y \prec f(x_1, \ldots, x_n)$. Pour une exécution σ, on a alors :

$$\sigma \vDash y \prec f(x_1, \ldots, x_n) \triangleq$$
$$\exists c \in \mathcal{C}(\mathbb{N}) : \forall i \in \mathbb{N} : y.\sigma_i = f(x_1.\sigma_{c(i)}, x_2.\sigma_{c(i)}, \ldots, x_n.\sigma_{c(i)})$$

La relation d'observation $y \prec f(x_1, \ldots, x_n)$ est aussi notée $y \prec f(X)$ avec $X = \langle x_1, \ldots, x_n \rangle$. Au long de l'exécution et grâce aux propriétés des horloges, on exprime que les valeurs de y sont définies par l'application de la fonction f aux valeurs prises par les variables du n-uplet $\langle x_1, \ldots, x_n \rangle$. L'observation introduit, par l'existence de l'horloge, un décalage logique modélisant le temps de calcul.

Les propriétés du calcul de f introduites par cette définition et dues aux propriétés des horloges sont le respect de l'ordre chronologique et une lecture synchrone des différentes entrées.

Il est possible de spécifier des calculs où la lecture des entrées n'est pas simultanée en définissant plusieurs relations d'observation. Par exemple :

- si on a une lecture de l'entrée x_1 avant x_2 alors on décrit le calcul par :
  $$y \prec f('x_1, x_2) \ \wedge \ 'x_1 \prec x_1$$
  l'observation introduite traduit le décalage entre la lecture de x_1 et celle de x_2 ;
- si on a une lecture décalée sans pouvoir définir un ordre alors on a :
  $$y \prec f('x_1, 'x_2) \ \wedge \ 'x_1 \prec x_1 \ \wedge \ 'x_2 \prec x_2$$

On donne une définition générale pour ces deux cas.

Remarque . *Ordre de lecture. Soit un calcul* $y = f(x_1, x_2, \ldots, x_n)$ *tel que les entrées* $x_1, x_2, \ldots, x_n$ *sont lues les unes après les autres. Alors le décalage introduit par l'ordre de lecture et par le calcul est exprimé par les relations d'observation suivantes :*

$$'x_1 \prec x_1$$
$$('' x_1, 'x_2) \prec ('x_1, x_2)$$
$$\ldots$$
$$('^{(n-1)} x_1, \ldots, 'x_{n-1}) \prec ('^{(n-2)} x_1, \ldots, x_{n-1})$$
$$y \prec f('^{(n-1)} x_1, \ldots, 'x_{n-1}, x_n)$$

Justification. Dans cette définition, chaque observation introduit un décalage entre la lecture de la i-ème et de la $(i+1)$-ème variable. Plus il existe de relations d'observations entre deux variables, plus le décalage est important. Par exemple, le décalage est plus important entre x_1 et y qu'entre x_2 et y et donc en chaque état la valeur de x_1 utilisée pour définir y est antérieure à celle de x_2.

Remarque . *Soit un calcul* $y = f(x_1, x_2, \ldots, x_n)$ *tel que les entrées* $x_1, x_2, \ldots, x_n$ *ne sont pas lues les unes après les autres ou de manière simultanée. On exprime le décalage entre les différentes lectures par les relations d'observation suivantes :*

$$
\begin{aligned}
&`x_1 \prec x_1\\
&`x_2 \prec x_2\\
&\ldots\\
&`x_n \prec x_n\\
&y \prec f(`x_1, `x_2, \ldots, `x_n)
\end{aligned}
$$

Justification. Ici, le décalage entre chacun des x_i et y est totalement indépendant des autres variables. Aucun ordre n'est donc défini.

Il est donc possible de modéliser des calculs avec trois ordres de lecture des entrées : synchronisé, ordonné ou sans ordre. D'autres ordres sont exprimables à l'aide de relations d'observation mais nous nous concentrons uniquement sur ces trois ordres, les plus communs.

3.2.3 *Architecture d'un système*

On décrit l'architecture d'un système via un ensemble d'observations correspondant à la définition 11 et représentant la propagation des valeurs de chaque variable dans le système. Il y a propagation à travers les communications et les différents calculs. Chaque communication du système sert à envoyer la copie de la valeur d'une variable d'un site à un autre. La communication de la valeur de x est spécifiée par une relation d'observation $`x \prec x$. Pour modéliser le calcul des valeurs de y à partir des valeurs de x en utilisant une fonction f, on définit une observation $y \prec f(x)$. Lors d'un calcul avec plusieurs entrées, une relation d'observation telle que $y \prec f(x_1, x_2)$ spécifie une lecture simultanée des entrées x_1 et x_2 pour le calcul de y. On peut aussi modéliser un calcul avec une lecture décalée des entrées par un ensemble d'observations.

Définition 12. *L'architecture d'un système et les liens entre les valeurs des différentes variables du système sont définis par un ensemble de relations d'observation* $\mathcal{O}bs$ *entre une variable et le résultat d'une fonction appliquée à un n-uplet de variables. Pour un tel ensemble, on note* $\mathcal{V}ar(\mathcal{O}bs)$ *l'ensemble des variables utilisées pour définir les relations d'observation de* $\mathcal{O}bs$*.*

Par exemple pour un ensemble de relations d'observation :

$$\mathcal{O}bs = \{y \prec x, z \prec f(y, t)\}$$

on a $\mathcal{V}ar(\mathcal{O}bs) = \{x, y, z, t\}$. L'architecture modélisée par $\mathcal{O}bs$ donne la propagation des valeurs de ces variables dans le système.

3.2.4 *Horloges d'une observation*

On s'intéresse aux décalages introduits par les calculs et les communications dans un système. On se concentre donc sur l'ensemble des horloges utilisées pour définir la relation d'observation.

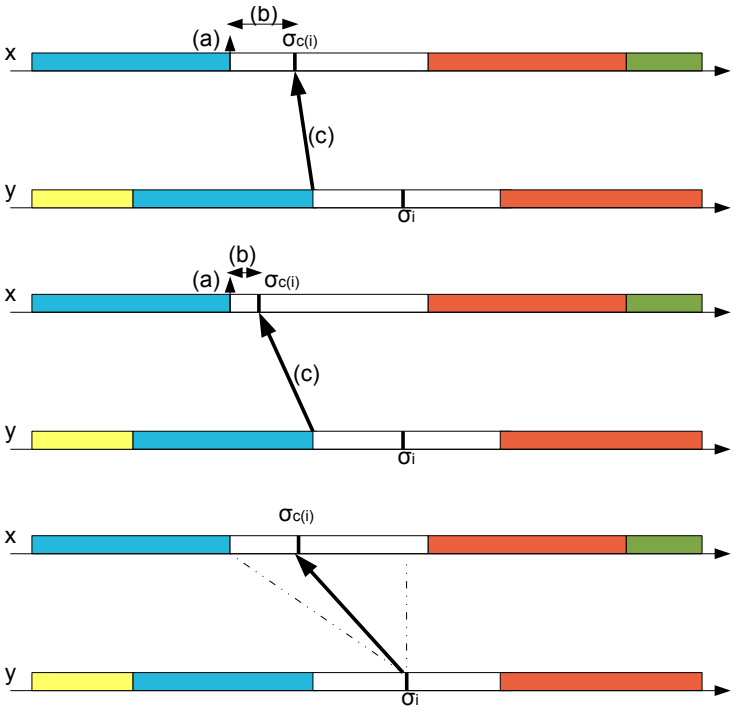

FIG. 6: Équivalence des horloges

Définition 13. *Horloges d'une observation. Soit une variable* y*, un n-uplet de variables* $X = \langle x_1, \ldots, x_n \rangle$ *et une fonction* f*. Alors pour une exécution* σ*, on définit* $\mathcal{C}(y \lessdot f(X)).\sigma$ *l'ensemble des horloges de* $\mathcal{C}(\mathbb{N})$ *tel que :*

$$\mathcal{C}(y \lessdot f(X)).\sigma \triangleq \{c \in \mathcal{C}(\mathbb{N}) | \forall i \in \mathbb{N} : y.\sigma_i = f(x_1.\sigma_{c(i)}, x_2.\sigma_{c(i)}, \ldots, x_n.\sigma_{c(i)})\}$$

Ce sont ces horloges qui caractérisent le décalage logique entre les sources et l'image au cours d'une exécution. On appelle $\mathcal{C}(y \lessdot f(X)).\sigma$ l'ensemble des horloges de l'observation $y \lessdot f(X)$ pour une exécution σ. Ces horloges ne représentent pas le mécanisme implanté pour réaliser l'observation. C'est une abstraction du décalage introduit et il existe plusieurs horloges d'observation pour une même exécution. La figure 6 illustre l'équivalence de plusieurs horloges pour une observation $y \lessdot x$. L'observation abstrait un envoi de message en trois étapes :

- (a) : mise à jour de x ;
- (b) : attente ;
- (c) : délivrance d'un message contenant la valeur de x à y.

L'horloge c de l'observation en un état σ_i peut alors se référer à l'état $\sigma_{c(i)}$ où le message a été créé. C'est ce qui est illustré dans les deux premiers cas de la figure 6. Dans ces deux cas, la durée des étapes b et c ne sont pas les mêmes et donc les états $\sigma_{c(i)}$ diffèrent. Cependant, dans les deux cas et pour l'état σ_i c'est la même valeur qui est assignée à y. Donc pour cet état, les deux comportements sont équivalents et on peut donc choisir indifféremment la valeur de $\sigma_{c(i)}$. Le dernier cas montre, entre les pointillés, l'ensemble des horloges possibles et donc l'ensemble des états $\sigma_{c(i)}$ possibles. Les états pointés par l'horloge d'observation doivent uniquement correspondre à la valeur de x à laquelle la valeur de y est liée dans l'état σ_i.

Les horloges permettent donc d'abstraire le mécanisme utilisé lors de la description du décalage entre la source et l'image. Du point de vue d'un état, tous les mécanismes ayant abouti à la même valeur de y dans l'état courant sont équivalents. Les différents états pointés par les horloges de l'observation correspondent aux différents états antérieurs où le mécanisme aurait pu lire la source. Chaque horloge d'observation représente un décalage logique possible introduit par l'observation. Pour une valeur de la source conservée pendant plusieurs états, plusieurs valeurs peuvent être données à ce décalage logique.

3.2.5 *Sémantique opérationnelle de la relation d'observation*

Nous expliquons comment la sémantique opérationnelle d'un système, plus précisément dans notre cas un système de transitions, peut être étendue pour prendre en compte la sémantique de la relation d'observation. Il s'agit donc de définir l'ensemble des exécutions d'un système qui vérifient une spécification étendue de relations d'observation. On s'appuie sur le travail réalisé par [Cha97] et accompagnant la définition originale de la relation d'observation.

Définition 14. *Sémantique opérationnelle de la relation d'observation. Soit un ensemble de relation d'observation $\mathcal{O}bs$ et une relation d'observation $y \prec f(X)$. On définit l'ensemble des exécutions $[\![\mathcal{O}bs]\!]_{\mathcal{O}}$ définies par la sémantique opérationnelle de la relation d'observation par :*

$$[\![\varnothing]\!]_{\mathcal{O}} \triangleq \Sigma$$
$$[\![\mathcal{O}bs \cup \{y \prec f(X)\}]\!]_{\mathcal{O}} \triangleq [\![\mathcal{O}bs]\!]_{\mathcal{O}} \cap \{\sigma | \sigma \vDash y \prec f(X)\}$$

où Σ est l'ensemble de toutes les exécutions infinies.

Soit un système de transitions S et $[\![S]\!]_{ST}$ l'ensemble des exécutions définies la sémantique de ce système de transitions. La sémantique de S étendu par un ensemble de relations d'observation $\mathcal{O}bs$ définit l'ensemble d'exécutions $[\![S, \mathcal{O}bs]\!]$ tel que

$$[\![S, \mathcal{O}bs]\!] \triangleq [\![S]\!]_{ST} \cap [\![\mathcal{O}bs]\!]_{\mathcal{O}}$$

La sémantique d'un système étendue est donc l'ensemble des exécutions définies par la spécification telles qu'en tout état les valeurs de l'image y de l'observation sont bien égales aux valeurs de l'expression $f(X)$ dans les états antérieurs. La relation d'observation n'introduit pas de nouveaux états dans une exécution.

Pour que la définition soit bien fondée, on étend si nécessaire les états des exécutions de la sémantique de S avec les variables de l'observation.

En pratique, les spécifications S qu'on étend avec des relations d'observation ne précisent pas les affectations de valeur de ces variables. Les valeurs de l'image dans la sémantique de la spécification étendue sont dans ce cas uniquement définies par la relation d'observation.

Pour l'analyse, on étudie de plus des systèmes où chaque variable est image d'au plus une relation d'observation.

Dans [Cha97], il s'agit d'un modèle opérationnel dans lequel une relation d'observation est implicitement satisfaite et donc toujours satisfiable. Cette satisfiabilité n'est pas vraie dans le cadre de la relation d'observation modélisant des calculs. Par exemple, une spécification définie par deux relations d'observation $y \prec (x + 1)$ et $x \prec (y + 1)$ n'est pas satisfiable car il n'existe pas d'état initial possible. Par la suite, on ne s'intéresse pas à un tel problème de satisfiabilité des relations d'observation. On ne s'intéresse en effet pas aux valeurs effectivement prises par les variables du système. On s'intéresse

uniquement aux comportements temporels des variables et aux liens, introduits par les relations d'observation, entre le comportement temporel des variables du système.

3.3 OUTILS POUR DÉFINIR LES PROPRIÉTÉS TEMPS RÉEL D'UN SYSTÈME

L'utilisation de relations d'observation permet de modéliser l'architecture d'un système. On donne des définitions complémentaires utilisée par la suite pour donner des propriétés temps réel sur une telle architecture.

3.3.1 *Profil temporel d'une variable*

On s'intéresse à la correction temporelle des valeurs prises par les variables du système. Cette correction ne dépend pas de la valeur prise par les variables mais dépend notamment des instants où cette valeur a été affectée. Afin de caractériser ces instants, on définit une variable, appelée profil temporel, qui décrit le comportement temporel d'une variable. Il s'agit de décrire les caractéristiques temporelles de l'historique des valeurs prises par cette variable lors d'une exécution.

Définition 15. *Profil temporel d'une variable. Soit une exécution séparée* σ *et une variable* x*. Les valeurs du profil temporel* $\hat{x}$ *de* x *sont définies par :*

$$\forall i \in \mathbb{N} : \hat{x}.\sigma_i \triangleq T.\sigma_{min\{j|\forall k \in [j..i]:\ x.\sigma_i = x.\sigma_k\}}$$

Les valeurs du profil temporel de x sont construites selon l'historique des valeurs de x. Dans chaque état, la valeur du profil temporel est définie par le dernier instant de changement de valeur de x c'est-à-dire par la valeur de la variable T dans l'état où la valeur actuelle est apparue et a, depuis, été gardée en continu. Un exemple de l'évolution d'une telle variable $\hat{x}$ est donnée figure 7. Les valeurs de $\hat{x}$ sont données selon les valeurs de T prises en chaque état. Lorsqu'il y a une mise à jour et un changement de la valeur de x comme à l'instant $T.\sigma_i$, la valeur de $\hat{x}$ est égale à celle de T. Dans les autres états, cette valeur n'est pas modifiée et est identique à celle à l'instant précédent. La valeur de $\hat{x}$ donne donc le dernier instant où la valeur de x a été modifiée. Par définition, pour toute variable x, la variable $\hat{x}$ est initialisée à 0.

Une première propriété est l'unicité du profil temporel pour une exécution et une variable x donné.

Proposition 5. *Unicité Soit une exécution séparée* σ *et une variable* x*. La définition du profil temporel* $\hat{x}$ *de* x *définit une variable* $\hat{x}$ *unique.*

Démonstration. Pour une exécution σ et une variable x, la définition du profil temporel définit une variable $\hat{x}$ unique si elle définit une valeur unique en chaque état de l'exécution. Pour chaque état σ_i, la valeur de $\hat{x}$ est défini par :

$$T.\sigma_{min\{j|\forall k \in [j..i]:\ x.\sigma_i = x.\sigma_k\}}$$

L'ensemble $\{j|\forall k \in [j..i] :\ x.\sigma_i = x.\sigma_k\}$ désigne l'ensemble des entiers j tel que pour tout état de l'exécution entre l'état σ_j et l'état σ_i la valeur de x est la même que dans l'état σ_i. Cet ensemble d'entier est minoré par 0 et donc il existe un minimum unique à cet ensemble. La valeur de la variable T dans l'état correspondant à ce minimum est unique, la valeur du profil temporel $\hat{x}$ est donc unique dans chaque état de l'exécution. La définition du profil temporel de x pour une exécution définit donc une variable $\hat{x}$ unique. □

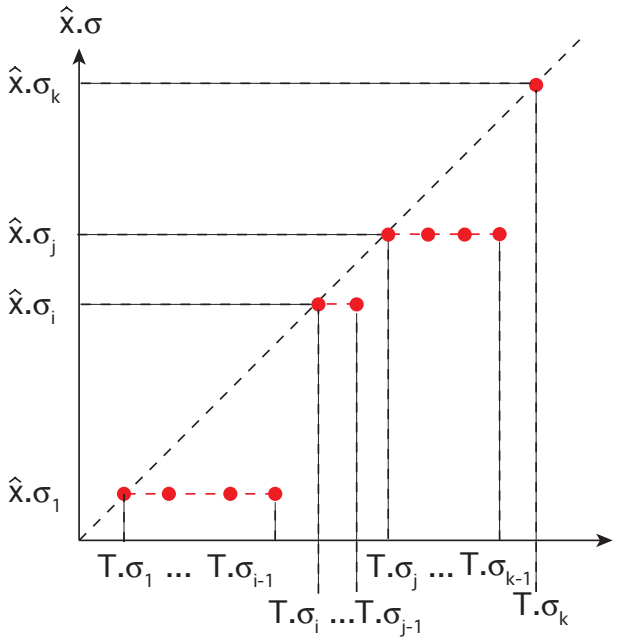

FIG. 7: Profil temporel $\hat{x}$ d'une variable x

Si dans deux états, les valeurs du profil temporel d'une variable x et donc les instants de changement de valeur sont les mêmes, alors on a la même valeur de x dans ces deux états.

Proposition 6. *Soit une variable x et une exécution séparée σ alors on a :*

$$\forall i, j \in \mathbb{N} : \hat{x}.\sigma_i = \hat{x}.\sigma_j \Rightarrow x.\sigma_i = x.\sigma_j$$

Démonstration. Soit une variable x et une exécution séparée σ. On suppose que l'on a pour deux entiers i et j :

$$\begin{array}{ll} & \hat{x}.\sigma_i = \hat{x}.\sigma_j \\ \Rightarrow & \quad \{\text{définition de } \hat{x}\} \\ & T.\sigma_{min\{l|\forall k\in[l..i]:\ x.\sigma_i=x.\sigma_k\}} = T.\sigma_{min\{l|\forall k\in[l..j]:\ x.\sigma_j=x.\sigma_k\}} \end{array}$$

Toute partie non vide de $\mathbb{N}$ a un minimum qui appartient à cette partie de $\mathbb{N}$. Comme i appartient à $\{l|\forall k \in [l..i] :\ x.\sigma_i = x.\sigma_k\}$, on a :

$$min\{l|\forall k \in [l..i] :\ x.\sigma_i = x.\sigma_k\} \in \{l|\forall k \in [l..i] :\ x.\sigma_i = x.\sigma_k\}$$

On a donc $x.\sigma_{min\{l|\forall k\in[l..i]:\ x.\sigma_i=x.\sigma_k\}} = x.\sigma_i$. De même, on a $x.\sigma_{min\{l|\forall k\in[l..j]:\ x.\sigma_j=x.\sigma_k\}} = x.\sigma_j$. On suppose que $x.\sigma_i \neq x.\sigma_j$, on a alors :

$$x.\sigma_{min\{l|\forall k\in[l..i]:\ x.\sigma_i=x.\sigma_k\}} \neq x.\sigma_{min\{l|\forall k\in[l..j]:\ x.\sigma_j=x.\sigma_k\}}$$

Or comme on a une exécution séparée, la valeur de T ne peut être la même dans ces deux états. On a donc une contradiction. On a donc nécessairement $x.\sigma_i = x.\sigma_j$. □

On peut aussi donner une définition différente du profil temporel avec une horloge sur les indices des états. En chaque état, la valeur de cette horloge pointe alors vers l'état dans lequel la valeur de la variable x a changé.

Définition 16. *État de changement de valeur. Soit une exécution séparée* σ *et une variable* x *vivace. L'horloge* $\hat{x}_c$ *est définie par :*

$$\forall i \in \mathbb{N} : \hat{x}_c(i) = \min\{j | \forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\}$$

On a besoin que la variable x soit vivace pour que l'horloge $\hat{x}_c$ soit bien une horloge et soit donc elle-même vivace.

Démonstration. On doit prouver que la définition de $\hat{x}_c$ définit bien une horloge.

1. Premièrement, par la définition, $\hat{x}_c$ ne dépasse pas son argument.

$$\begin{array}{ll} & \forall i \in \mathbb{N} : x.\sigma_i = x.\sigma_i \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & \forall i \in \mathbb{N} : \forall k \in [i..i] : \ x.\sigma_i = x.\sigma_k \\ \Rightarrow & \quad \{\text{définition de l'ensemble}\} \\ & \forall i \in \mathbb{N} : i \in \{j | \forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\} \\ \Rightarrow & \quad \{\text{définition du minimum}\} \\ & \forall i \in \mathbb{N} : \min\{j | \forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\} \leq i \\ \Rightarrow & \quad \{\text{par définition}\} \\ & \forall i \in \mathbb{N} : \hat{x}_c(i) \leq i \end{array}$$

2. Soit $i \in \mathbb{N}$, concernant la croissance, on détermine deux cas entre σ_i et σ_{i+1}.

2.A. Si on a $x.\sigma_i \neq x.\sigma_{i+1}$ alors on a :

$$\begin{array}{ll} & x.\sigma_i \neq x.\sigma_{i+1} \\ \Rightarrow & \quad \{\text{définition de l'ensemble}\} \\ & i \notin \{j | \forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k\} \\ \Rightarrow & \quad \{\text{propriétés des entiers}\} \\ & \{j | \forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k\} = \{i+1\} \\ \Rightarrow & \quad \{\text{minimum d'un ensemble singleton}\} \\ & \hat{x}_c(i+1) = i+1 \end{array}$$

or $\forall i \in \mathbb{N} : \hat{x}_c(i) \leq i$ donc $\hat{x}_c(i) < \hat{x}_c(i+1)$.

2.B. Si on a $x.\sigma_i = x.\sigma_{i+1}$ alors :

$$\begin{array}{ll} & \forall k \in \mathbb{N} : \ x.\sigma_i = x.\sigma_k \Leftrightarrow x.\sigma_{i+1} = x.\sigma_k \\ \Rightarrow & \quad \{\text{application à un sous-intervalle de } \mathbb{N}\} \\ & \forall j \in \mathbb{N} : \forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k \Leftrightarrow x.\sigma_{i+1} = x.\sigma_k \\ \Rightarrow & \quad \{\text{propriété du } \forall\} \\ & \forall j \in \mathbb{N} : (\forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k) \Leftrightarrow (\forall k \in [j..i] : \ x.\sigma_{i+1} = x.\sigma_k) \\ \Rightarrow & \quad \{\text{application à un sous-intervalle de } \mathbb{N}\} \\ & \forall j \in [0..i] : (\forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k) \Leftrightarrow (\forall k \in [j..i] : \ x.\sigma_{i+1} = x.\sigma_k) \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & \forall j \in [0..i] : (\forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k) \Leftrightarrow (\forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k) \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & \forall j \in [0..i] : j \in \{j | \forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\} \Leftrightarrow j \in \{j | \forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k\} \end{array}$$

Or on a $\forall l > i+1 : l \notin \{j | \forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k\}$ car alors $[l..i+1]$ est un intervalle vide. Les éléments entre 0 et i qui appartiennent à un des deux ensembles appartiennent

à l'autre, les éléments supérieurs à $i+1$ n'appartiennent à aucun des deux ensembles. On a donc :

$$\{j|\forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\} \cup \{i+1\} = \{j|\forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k\}$$

Étant donné que

$$min(\{j|\forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\}) < i+1$$

alors :

$$min(\{j|\forall k \in [j..i] : \ x.\sigma_i = x.\sigma_k\}) = min(\{j|\forall k \in [j..i+1] : \ x.\sigma_{i+1} = x.\sigma_k\})$$

donc si $x.\sigma_i = x.\sigma_{i+1}$ alors $\hat{x}_c(i) = \hat{x}_c(i+1)$ et donc $\hat{x}_c$ est croissante.

3. Concernant la vivacité, on a prouvé que si $x.\sigma_i \neq x.\sigma_{i+1}$ alors $\hat{x}_c(i) < \hat{x}_c(i+1)$. Comme x est vivace, alors $\hat{x}_c$ est aussi vivace. Donc $\hat{x}_c$ est une horloge. □

On a quasiment prouvé, dans la preuve précédente, la proposition suivante :

Proposition 7. *Soit une exécution séparée σ et une variable x. Alors on a la propriété suivante :*

$$\forall d, f \in \mathbb{N} : d < f : \left(\begin{array}{c} \forall i \in [d..f] : x.\sigma_i = x.\sigma_d \\ \Leftrightarrow \\ \forall i \in [d..f] : \hat{x}_c(i) = \hat{x}_c(d) \end{array} \right)$$

Démonstration. Pour une exécution séparée et une variable x, on a déjà démontré les deux propriétés suivantes :

$$\begin{array}{ll} & \left(\begin{array}{l} x.\sigma_i = x.\sigma_{i+1} \Rightarrow \hat{x}_c(i) = \hat{x}_c(i+1) \\ x.\sigma_i \neq x.\sigma_{i+1} \Rightarrow \hat{x}_c(i) \neq \hat{x}_c(i+1) \end{array} \right) \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & x.\sigma_i = x.\sigma_{i+1} \Leftrightarrow \hat{x}_c(i) = \hat{x}_c(i+1) \end{array}$$

La preuve sur un intervalle s'obtient simplement en considérant tous les voisins à partir de d. □

Les deux définitions des variables relatives aux changements de valeurs d'une variable sont liées par la proposition suivante :

Proposition 8. *Équivalence des définitions. Soit une exécution séparée σ et une variable x. Alors on a :*

$$\forall i \in \mathbb{N} : \hat{x}.\sigma_i = T.\sigma_{\hat{x}_c(i)}$$

Cette équivalence est déduite directement des définitions des deux variables. On en déduit la propriété suivante :

Proposition 9. *Soit une exécution séparée σ et une variable vivace x, alors la variable $\hat{x}$ est croissante.*

Démonstration. D'après la proposition 8 et sachant que $\hat{x}_c$ et T sont croissantes, on en déduit que $\hat{x}$ est croissante. □

On s'intéresse au comportement temps réel des variables, la valeur effectivement prise par une variable n'est pas l'objet de notre étude. $\hat{\hat{x}}_c$ et $\hat{\hat{x}}$ permettent de caractériser le comportement temporel de la variable x, soit en terme de temps logique ; soit en terme de temps réel, abstraction faite du comportement de x en termes d'états et de valeurs.

Quand une variable x est mise à jour avec une nouvelle valeur, la valeur de $\hat{x}$ est aussi mise à jour avec l'instant courant. On a donc la propriété suivante :

Proposition 10. *Soit une exécution séparée σ et une variable vivace x. Alors, on a la propriété suivante*

$$\forall i \in \mathbb{N} : x.\sigma_i = x.\sigma_{i+1} \Leftrightarrow \hat{x}.\sigma_i = \hat{x}.\sigma_{i+1}$$

Démonstration. Soit une exécution séparée σ et une variable x et soit i un entier tel que :

$$\begin{array}{ll} & x.\sigma_i = x.\sigma_{i+1} \\ \Leftrightarrow & \quad \{\text{proposition 7 sur l'intervalle } [i..i+1]\} \\ & \hat{x}_c(i) = \hat{x}_c(i+1) \\ \Leftrightarrow & \quad \{\text{unicité de la valeur d'une variable dans un état}\} \\ & T.\sigma_{\hat{x}_c(i)} = T.\sigma_{\hat{x}_c(i+1)} \\ \Leftrightarrow & \quad \{\text{proposition 8}\} \\ & \hat{x}.\sigma_i = \hat{x}.\sigma_{i+1} \end{array}$$

□

Une deuxième variable est introduite pour quantifier la stabilité d'une variable c'est-à-dire le temps entre deux changements de valeurs. La valeur de cette variable en un état est donc la durée pendant laquelle la valeur courante de la variable est continûment gardée par la variable autour de cet état.

Définition 17. *Stabilité. Soit une exécution séparée σ et une variable x. La stabilité d_x de la variable x est définie par :*

$$\forall i \in \mathbb{N} : d_x.\sigma_i \triangleq \begin{cases} +\infty \textit{ si } \forall k \geq i : x.\sigma_k = x.\sigma_i \\ T.\sigma_{min\{j | j \geq i \wedge x.\sigma_i \neq x.\sigma_j\}} - \hat{x}.\sigma_i \textit{ sinon} \end{cases}$$

Si jamais dans une exécution, à partir d'un état σ_i la variable ne change plus de valeur, alors la valeur de la stabilité sera $+\infty$ pour σ_i et les états suivants. On a alors la propriété suivante :

Proposition . *Soit une exécution séparée σ et une variable x vivace. Alors :*

$$\forall i \in \mathbb{N} : d_x.\sigma_i \neq +\infty$$

Démonstration. Si une variable est vivace, pour tout état il existe un état futur avec une nouvelle valeur de cette variable. En reprenant la définition de la durée entre deux changements de valeurs, il n'est donc pas possible que l'ensemble $\{j | \forall k \in [i..j[: x.\sigma_i = x.\sigma_k \wedge x.\sigma_i \neq x.\sigma_j\}$ soit vide. □

Même si la durée entre deux changements de valeur est nécessairement finie, il n'existe cependant pas de borne sur cette durée. On a aussi la propriété suivante :

Proposition 11. *Soit une exécution séparée σ et une variable x. Alors :*

$$\forall i \in \mathbb{N} : d_x.\sigma_i \neq +\infty \Rightarrow d_x.\sigma_i + \hat{x}.\sigma_i = min\{\hat{x}.\sigma_j | j \in \mathbb{N} \wedge \hat{x}.\sigma_i < \hat{x}.\sigma_j\}$$

La valeur de $d_x.\sigma_i + \hat{x}.\sigma_i$ donne donc la date du prochain changement de valeur de x et donc la prochaine valeur du profil temporel différente de la valeur actuelle.

Démonstration. Soit une exécution séparée σ et une variable x. Soit un entier i tel que $d_x.\sigma_i \neq +\infty$. Alors on a :

$$d_x.\sigma_i + \hat{x}.\sigma_i = T.\sigma_{min\{j|j\geq i \wedge x.\sigma_i \neq x.\sigma_j\}}$$

Soit $l = min\{j|j \geq i \wedge x.\sigma_i \neq x.\sigma_j\}$. On a $x.\sigma_{l-1} \neq x.\sigma_l$. Pour l'état précédant l, on a donc une valeur différente de x et donc :

$$l = min\{j|\forall k \in [j..l] : x.\sigma_k = x.\sigma_l\}$$

Or on sait que pour tout i :

$$\hat{x}.\sigma_i = T.\sigma_{min\{j|\forall k\in[j..i]:\ x.\sigma_i=x.\sigma_k\}}$$

donc appliqué à l, on a $\hat{x}.\sigma_l = T.\sigma_l$. Soit :

$$\begin{array}{ll} & d_x.\sigma_i + \hat{x}.\sigma_i = \hat{x}.\sigma_{min\{j|j\geq i \wedge x.\sigma_i\neq x.\sigma_j\}} \\ \Rightarrow & \quad \{\text{proposition 10}\} \\ & d_x.\sigma_i + \hat{x}.\sigma_i = \hat{x}.\sigma_{min\{j|j\geq i \wedge \hat{x}.\sigma_i\neq \hat{x}.\sigma_j\}} \end{array}$$

Les éléments j de $\{j|j \geq i \wedge \hat{x}.\sigma_i \neq \hat{x}.\sigma_j\}$ sont supérieurs à i et ont une valeur de $\hat{x}.\sigma_j$ différente de $\hat{x}.\sigma_i$. Comme $\hat{x}$ est croissant, il s'agit des entiers j tels que $\hat{x}.\sigma_j > \hat{x}.\sigma_i$. Donc on a :

$$\begin{array}{ll} & d_x.\sigma_i + \hat{x}.\sigma_i = \hat{x}.\sigma_{min\{j|j\in\mathbb{N} \wedge \hat{x}.\sigma_i<\hat{x}.\sigma_j\}} \\ \Rightarrow & \quad \{\hat{x} \text{ croissante sur les états}\} \\ & d_x.\sigma_i + \hat{x}.\sigma_i = min\{\hat{x}.\sigma_j|j \in \mathbb{N} \wedge \hat{x}.\sigma_i < \hat{x}.\sigma_j\} \end{array}$$

□

3.3.2 *Occurrences*

Le profil temporel d'une variable a été introduit afin de caractériser le comportement temporel d'une variable. Les valeurs du profil temporel d'une variable x dans une exécution sont définies par rapport aux valeurs prises par x dans cette exécution. Notamment, la valeur du profil temporel $\hat{x}$ d'une variable x est modifiée lorsque la valeur de x est modifiée.

Il est possible que cette variable x soit mise à jour sans que sa valeur soit modifiée. On peut alors considérer que malgré le fait que la valeur de x n'a pas changé, l'événement de mise à jour est important. Par exemple pour l'étude de l'évolution de l'environnement, supposons qu'on veut toujours avoir une mesure récente d'un phénomène extérieur. L'environnement peut ne pas évoluer entre deux mesures mais la deuxième mesure, même si elle ne modifie pas la valeur, est importante car elle justifie que la valeur reste récente indépendamment de l'évolution de l'environnement.

Dans le cas d'un calcul $y = f(X)$, un problème similaire se pose. Si l'on considère le calcul $y = \lfloor \frac{x}{2} \rfloor$ (partie entière de $\frac{x}{2}$) alors deux valeurs différentes de x peuvent donner une même valeur de y. Si la validité temporelle des valeurs de y dépend des valeurs de x utilisées en entrée, par exemple de leur fraîcheur, alors l'utilisation d'une nouvelle valeur de x valide la valeur de y même si celle-ci n'a pas changé. Bien que la mise

à jour n'entraîne pas de changement de valeur, elle est importante pour caractériser le comportement temporel de l'image. Inversement, si c'est la même occurrence de x qui est utilisée pour effectuer le calcul, l'occurrence de y obtenue est identique à la précédente et le calcul n'est pas pertinent. On a un comportement similaire pour les occurrences d'un n-uplet de variables qui ne sont modifiées que si l'occurrence d'une des variables du n-uplet est modifiée.

La définition du profil temporel sur l'historique des valeurs d'une variable ne permet donc pas de modéliser parfaitement le comportement temporel d'une variable. Pour pouvoir intégrer ces instants de mise à jour dans le profil temporel d'une variable, on introduit la notion d'occurrence. On utilise alors un compteur des mises à jour d'une expression qui permet de différencier deux états où les valeurs prises par une expression sont identiques mais issues de deux mises à jour différentes. Si lors d'une transition la valeur d'une expression est modifiée, il y a nécessairement eu une mise à jour et le compteur croît. Le compteur peut aussi croître alors que la valeur de l'expression associée n'est pas modifiée. On a alors un événement de mise à jour n'ayant pas modifié la valeur de cette expression mais impliquant un changement d'occurrence.

On définit le compteur d'occurrences pour différentes expressions :

Définition 18. *Compteur d'occurrences.*

- *Compteur d'occurrence d'une variable : on définit un compteur d'occurrences d'une variable x comme une variable croissante occ_x tel que dans une exécution σ :*

$$\forall i \in \mathbb{N} : \mathrm{occ}_x.\sigma_{i+1} \geq \mathrm{occ}_x.\sigma_i \ \wedge \ (x.\sigma_{i+1} \neq x.\sigma_i \Rightarrow \mathrm{occ}_x.\sigma_{i+1} \neq \mathrm{occ}_x.\sigma_i)$$

- *Compteur d'occurrence d'un n-uplet de variables : un compteur d'occurrence d'un n-uplet $X = \langle x_1, x_2, \ldots, x_n \rangle$ de variable est défini comme une variable croissante occ_X tel que dans une exécution σ :*

$$\forall i \in \mathbb{N} : \mathrm{occ}_X.\sigma_{i+1} \geq \mathrm{occ}_X.\sigma_i \wedge (\mathrm{occ}_X.\sigma_{i+1} \neq \mathrm{occ}_X.\sigma_i \Leftrightarrow \exists k \in [1..n] : \mathrm{occ}_{x_k}.\sigma_{i+1} \neq \mathrm{occ}_{x_k}.\sigma_i)$$

  où les variables occ_{x_k} sont les compteurs d'occurrences des variables x_k.

- *Compteur d'occurrence du résultat d'une fonction : pour une fonction f et un n-uplet de variables X, un compteur d'occurrences de l'expression $f(X)$ est défini comme une variable croissante $\mathrm{occ}_{f(X)}$ tel que dans une exécution σ :*

$$\forall i \in \mathbb{N} : \mathrm{occ}_{f(X)}.\sigma_i = \mathrm{occ}_X.\sigma_i$$

  où occ_X est un compteur d'occurrences du n-uplet de variables X.

Ces compteurs sont utilisés pour définir les occurrences d'une expression :

Définition 19. *Occurrences d'une expression. Pour une expression e, on définit les occurrences de l'expression e, noté $\bar{e}$, comme un couple tel que pour toute exécution σ, on a :*

$$\forall i \in \mathbb{N} : \bar{e}.\sigma_i \triangleq \langle e.\sigma_i, \mathrm{occ}_e.\sigma_i \rangle$$

Dans le cadre d'une observation, on veut lier l'évolution des occurrences de l'image aux évolutions des occurrences des sources. On veut que l'occurrence de l'image évolue uniquement lorsqu'elle est construite à partir de nouvelles occurrences des sources. On considère alors que refaire le calcul ou la communication modélisé par une observation avec les mêmes occurrences des sources n'intervient pas sur la validité temporelle du résultat et n'entraîne donc pas de changement d'occurrence. En d'autres termes, il est inutile de refaire un calcul en utilisant les mêmes occurrences des entrées.

Pour modéliser ce lien entre les occurrences des sources et l'occurrence de l'image, on définit l'observation des occurrences.

Définition 20. *Observation des occurrences. Soit une variable* y*, une fonction* f *et un n-uplet de variables* $X = \langle x_1, x_2, \ldots, x_n \rangle$*. la variable* y *est une observation* $y \trianglelefteq f(X)$ *des occurrences des variables de* X *par* f *si on a :*

$$\sigma \vDash y \trianglelefteq f(X) \triangleq \sigma \vDash \bar{y} \prec f(\bar{X})$$

Le compteur d'occurrence de l'image y est donc une observation du compteur d'occurrence de l'expression $f(X)$ observée. Si y est mis à jour, le compteur d'occurrence de y change et croît. Le compteur de l'expression source $f(X)$ observée a donc cru précédemment et donc d'après la définition de ce compteur, un des éléments de X a été mis à jour. y est donc liée par l'horloge d'observation choisie à une nouvelle occurrence d'une des sources.

Pour justifier le comportement du compteur d'occurrence de l'image d'une observation des occurrences, on prend comme exemple un système réactif uniquement décrit par son architecture sous la forme d'un ensemble de relations d'observation. Une variable qui n'est pas image d'une observation n'est pas une valeur produite à partir d'autres variables du système. Elle modélise alors une mesure de l'environnement et donc une des entrées du système. Il est alors important de noter que la valeur de cette variable a été remise à jour et qu'elle suit bien les évolutions de l'environnement même si sa valeur n'a pas changé. Comme l'architecture ne modélise pas les causes des mises à jour et sur quoi sont basées les valeurs de cette variable, cela doit être explicitement donné en incrémentant le compteur d'occurrences.

Dans le cas où l'on considère l'image d'une observation, le compteur d'occurrences ne peut évoluer que si le compteur d'occurrences de l'expression observée évolue. Alors les compteurs d'occurrences des sources dans les états pointés par l'horloge d'observation évoluent aussi et donc la valeur de l'image est basée sur de nouvelles occurrences des sources. Ici, l'architecture modélise les liens entre une variable image et les autres variables et donc justifie les valeurs qui sont prises par cette image. On n'autorise pas les évolutions du compteur d'occurrences de cette image sans que celui d'une des sources n'évolue. Alors, faire un calcul avec les mêmes occurrences des sources ou communiquer deux fois la même occurrence d'une source est inutile et ne génère pas une nouvelle occurrence de l'image. En effet, en remontant aux sources des différentes observations, l'image est liée aux mêmes occurrences des entrées du système et aux mêmes mesures de l'environnement et effectuer un tel calcul ou une telle communication n'est pas pertinent car ne génère pas une nouvelle valeur plus fraîche.

A travers les définitions des compteurs d'occurrences des variables du système, on étudie la propagation des différentes occurrences dans le système. On ne se focalise alors pas sur les valeurs mais sur la propagation des nouvelles entrées. On a la propriété suivante :

Proposition 12. *Pour une variable* x *et pour tout exécution* σ*, on a :*

$$\forall i, j \in \mathbb{N} : \bar{x}.\sigma_i = \bar{x}.\sigma_j \Leftrightarrow occ_x.\sigma_i = occ_x.\sigma_j$$

On en déduit que le changement d'occurrence est défini par l'évolution du compteur d'occurrences.

Démonstration. On prouve tout d'abord le lemme suivant :

Lemme . *Pour une variable* x *et pour toute exécution* σ*, on a :*

$$\forall i, j \in \mathbb{N} : occ_x.\sigma_i = occ_x.\sigma_j \Rightarrow x.\sigma_i = x.\sigma_j$$

Soit une variable x, une exécution σ et deux entiers i, j tel que $i \leq j$. On suppose que l'on a : $occ_x.\sigma_i = occ_x.\sigma_j$. Alors comme occ_x est croissante, on a :

$$\forall k \in [i..j] : occ_x.\sigma_k = occ_x.\sigma_i$$

On sait par définition de occ_x que si $occ_x.\sigma_i = occ_x.\sigma_{i+1}$ alors $x.\sigma_i = x.\sigma_{i+1}$. Par récurrence simple entre i et j, on prouve donc que :

$$\forall k \in [i..j] : x.\sigma_k = x.\sigma_i$$

Démontrons maintenant la proposition principale. Soit une variable x, une exécution σ et deux entiers i, j. On suppose que l'on a :

$$\begin{array}{ll} & \bar{x}.\sigma_i = \bar{x}.\sigma_j \\ \Leftrightarrow & \quad \{\text{par définition}\} \\ & \langle x.\sigma_i, occ_x.\sigma_i \rangle = \langle x.\sigma_j, occ_x.\sigma_j \rangle \\ \Leftrightarrow & \quad \{\text{définition d'un couple}\} \\ & (x.\sigma_i = x.\sigma_j) \wedge (occ_x.\sigma_i = occ_x.\sigma_j) \end{array}$$

Or $occ_x.\sigma_i = occ_x.\sigma_j \Rightarrow x.\sigma_i = x.\sigma_j$ donc on a bien :

$$\begin{array}{ll} & (x.\sigma_i = x.\sigma_j) \wedge (occ_x.\sigma_i = occ_x.\sigma_j) \\ \Leftrightarrow & \quad \{\text{propriété logique}\} \\ & occ_x.\sigma_i = occ_x.\sigma_j \end{array}$$

□

Les occurrences ont été introduites pour compléter la définition du profil temporel d'une variable et permettre de mieux décrire le comportement temporel d'une variable. Pour compléter cette définition, on définit donc le profil temporel des occurrences d'une variable.

Définition 21. *Profil temporel des occurrences d'une variable. Pour une variable x, on définit le profil temporel des occurrences de la variable x, noté $\hat{\bar{x}}$, tel que pour toute exécution σ, on a :*

$$\forall i \in \mathbb{N} : \hat{\bar{x}}.\sigma_i \triangleq \hat{\bar{x}}.\sigma_i$$

Les valeurs du profil temporel des occurrences de x sont des instants de changement de valeur de la variable occurrence $\bar{x}$ de x. Ces instants sont donc les instants de mise à jour de x. Cette variable permet d'étudier les propriétés temps réel du système en caractérisant le comportement temporel de chacune de ces variables et en tenant compte de toutes les mises à jours significatives. Ainsi on se concentre donc non pas sur les valeurs mais sur le comportement temporel des variables.

On donne aussi une nouvelle définition de la variable stabilité dans le cadre des occurrences.

Définition 22. *Stabilité des occurrences. Soit une exécution séparée σ et une variable x. La stabilité $d_{\bar{x}}$ des occurrences est définie par :*

$$\forall i \in \mathbb{N} : d_{\bar{x}}.\sigma_i = T.\sigma_{min\{j | j \geq i \wedge \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} - \hat{\bar{x}}.\sigma_i$$

Les valeurs prises par la variable stabilité définie ici sont pour chaque état la durée entre la dernière mise à jour et la prochaine mise à jour.

Donc dans le cas d'une variable qui prend une nouvelle valeur à chaque mise à jour, on a alors $\hat{\bar{x}} = \hat{x}$ étant donné que $\hat{x}$ est modifiée avec occ_x. Dans le cas où chaque mise à jour n'implique pas un changement de valeur, il faut spécifier explicitement les instants où x est mise à jour en utilisant la variable occ_x. La définition de $\hat{\bar{x}}$ donne les propriétés suivantes :

Proposition 13. *Soit une variable x et une exécution σ alors on a :*

$$\forall i, j \in \mathbb{N} : \bar{x}.\sigma_i = \bar{x}.\sigma_j \Leftrightarrow \hat{\bar{x}}.\sigma_i = \hat{\bar{x}}.\sigma_j$$

Démonstration. Il s'agit d'une application directe de la proposition 10. □

On en déduit de plus la propriété suivante :

Proposition 14. *Soit une variable x et une exécution σ alors on a :*

$$\forall i, j \in \mathbb{N} : \hat{\bar{x}}.\sigma_i = \hat{\bar{x}}.\sigma_j \Rightarrow x.\sigma_i = x.\sigma_j$$

Il s'agit de préciser que l'on a bien la même valeur de la variable x dans deux états ayant la même date de mise à jour.

Démonstration. Soit une variable x, une exécution σ et deux entiers i et j, alors on a :

$\hat{\bar{x}}.\sigma_i = \hat{\bar{x}}.\sigma_j$
$\Rightarrow$ {proposition 13}
$\bar{x}.\sigma_i = \bar{x}.\sigma_j$
$\Rightarrow$ {par définition}
$\langle x.\sigma_i, occ_x.\sigma_i \rangle = \langle x.\sigma_j, occ_x.\sigma_j \rangle$
$\Rightarrow$ {décomposition du couple}
$x.\sigma_i = x.\sigma_j$

□

Par la suite on s'intéressera uniquement aux observations d'occurrences et donc aux propriétés de la propagation des occurrences. Pour cela, on définit l'ensemble des horloges d'une observation des occurrences.

Définition 23. *Horloges d'une observation des occurrences. Soit une observation des occurrences $y \bar{\prec} f(X)$ avec $X = \langle x_1, x_2, \ldots, x_n \rangle$ et une exécution σ vérifiant cette observation. On note $\mathcal{C}(y \bar{\prec} f(X)).\sigma$ l'ensemble des horloges de $\mathcal{C}(\bar{y} \prec f(\bar{X})).\sigma$.*

Les horloges de l'observation des occurrences indiquent avec quelles occurrences des sources sont faites les mises à jour de l'image.
On a la propriété suivante :

Proposition 15. *Soit une observation des occurrences $y \bar{\prec} f(X)$ avec $X = \langle x_1 \ldots x_n \rangle$ et une exécution σ vérifiant cette observation, alors on a :*

$$\forall i, j \in \mathbb{N} : \forall c \in \mathcal{C}(y \bar{\prec} f(X)).\sigma : (\forall x_k \in X : \bar{x_k}.\sigma_{c(i)} = \bar{x_k}.\sigma_{c(j)}) \Leftrightarrow \bar{y}.\sigma_i = \bar{y}.\sigma_j$$

et :

$$\forall i, j \in \mathbb{N} : \forall c \in \mathcal{C}(y \bar{\prec} f(X)).\sigma : (\forall x_k \in X : occ_{x_k}.\sigma_{c(i)} = occ_{x_k}.\sigma_{c(j)}) \Leftrightarrow occ_y.\sigma_i = occ_y.\sigma_j$$

et finalement :

$$\forall i, j \in \mathbb{N} : \forall c \in \mathcal{C}(y \bar{\prec} f(X)).\sigma : (\forall x_k \in X : \hat{\bar{x}}_k.\sigma_{c(i)} = \hat{\bar{x}}_k.\sigma_{c(j)}) \Leftrightarrow \hat{\bar{y}}.\sigma_i = \hat{\bar{y}}.\sigma_j$$

On a déjà vu que l'évolution des occurrences de l'image dépend des sources mais aussi de l'horloge de l'observation qui est utilisée. En utilisant l'observation des occurrences, on utilise uniquement le sous ensemble des horloges de l'observation des valeurs qui correspond à l'évolution des occurrences de l'image. Alors cette proposition démontre le lien existant entre les occurrences des sources et les occurrences des images.

Démonstration. Soit une observation des occurrences $y \preccurlyeq f(X)$ avec $X = \langle x_1 \dots x_n \rangle$ et une exécution σ vérifiant cette observation. Il suffit de prouver la première partie de la proposition, la suite découlant des équivalences prouvées par les propositions 13 et 12. On suppose que l'on a deux états i, j tel que :

$$
\begin{aligned}
& \bar{y}.\sigma_i = \bar{y}.\sigma_j \\
\Leftrightarrow \quad & \{\text{proposition 12}\} \\
& occ_y.\sigma_i = occ_y.\sigma_j \\
\Leftrightarrow \quad & \{\text{définition de l'observation des occurrences et des horloges d'une observation}\} \\
& \forall c \in \mathcal{C}(y \preccurlyeq f(X)).\sigma : occ_{f(X)}.\sigma_{c(i)} = occ_{f(X)}.\sigma_{c(j)} \\
\Leftrightarrow \quad & \{\text{définition du compteur des occurrences de } f(X)\} \\
& \forall c \in \mathcal{C}(y \preccurlyeq f(X)).\sigma : occ_X.\sigma_{c(i)} = occ_X.\sigma_{c(j)} \\
\Leftrightarrow \quad & \{\text{définition du compteur des occurrences de } X\} \\
& \forall c \in \mathcal{C}(y \preccurlyeq f(X)).\sigma : \forall x_k \in X : occ_{x_k}.\sigma_{c(i)} = occ_{x_k}.\sigma_{c(j)}) \\
\Leftrightarrow \quad & \{\text{proposition 12}\} \\
& \forall c \in \mathcal{C}(y \preccurlyeq f(X)).\sigma : \forall x_k \in X : \bar{x_k}.\sigma_{c(i)} = \bar{x_k}.\sigma_{c(j)})
\end{aligned}
$$

□

3.3.3 *Chemins de propagation*

Chaque relation d'observation définit un lien de dépendance entre les valeurs des sources et celles de l'image. Même si deux variables du système ne sont pas directement liées par une relation d'observation, l'architecture du système peut créer un lien entre les valeurs de ces variables. Si l'on a deux observations $`x \preccurlyeq x$ et $y \preccurlyeq f(`x)$ modélisant une communication entre $`x$ et $x$ et un calcul des valeurs de $y$ basé sur les valeurs de $`x$ alors les valeurs de y sont liées à celles de x. On veut pouvoir définir des propriétés temporelles sur la propagation des occurrences par chaque observation mais aussi sur la propagation des occurrences à travers toute l'architecture. On cherche donc à caractériser le lien entre y et x dans notre exemple.

Proposition 16. *Un ensemble d'observations $\bar{\mathcal{O}}bs des occurrences forme un graphe orienté :*
- *les variables $\mathcal{V}ar(\bar{\mathcal{O}}bs)$ sont les nœuds ou sommets du graphe*
- *un arc $x \to y$ d'un nœud x vers un nœud y existe si :*

$$\exists X : x \in X \ \wedge \ \exists f : (y \preccurlyeq f(X)) \in \bar{\mathcal{O}}bs$$

Dans un tel graphe, deux variables peuvent être liées par plusieurs chemins, par exemple si l'on a les observations suivantes représentées figure 8 :

$$`x \preccurlyeq x; ``x \preccurlyeq x; y_1 \preccurlyeq f_1(`x), y_2 \preccurlyeq f_2(``x), `y_1 \preccurlyeq y_1, `y_2 \preccurlyeq y_2, z \preccurlyeq g(`y_1, `y_2)$$

Ces observations peuvent représenter des communications de x à deux sites différents utilisant ces valeurs pour calculer les valeurs de y_1 et y_2. Ces valeurs sont elles-mêmes communiquées à un autre site et utilisées pour calculer z. Alors il y a deux chemins liant x à z. Le décalage le long de ces deux chemins n'est pas le même, par exemple si

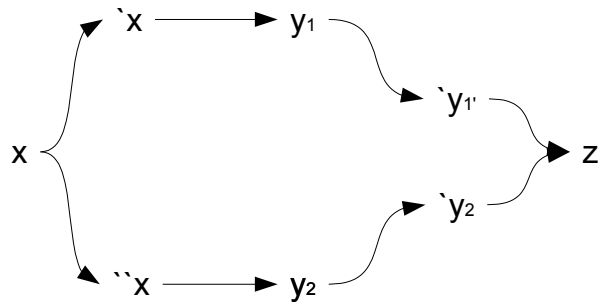

FIG. 8: Multiplicité des chemins entre deux variables

les propriétés des communications mises en oeuvre diffèrent. On veut donc pouvoir les dissocier.

Définition 24. *Chemin entre deux variables. Étant donné un ensemble de relations d'observation d'occurrences* $\bar{\mathcal{O}}\mathrm{bs}$ *définissant un graphe orienté, on définit un chemin de propagation par une séquence de variables* $[x_0, x_1, \ldots, x_n]$ *de* $\mathcal{V}\mathrm{ar}(\bar{\mathcal{O}}\mathrm{bs})$*. Ce chemin est bien défini par* $\bar{\mathcal{O}}\mathrm{bs}$ *si :*

$$\forall i \in [0..n-1], \exists X_i : x_i \in X_i \ \wedge \ \exists f : (x_{i+1} \preccurlyeq f(X_i)) \in \bar{\mathcal{O}}\mathrm{bs}$$

Soit pour chaque $i \in [0..n-1]$ *et dans le graphe défini par* $\bar{\mathcal{O}}\mathrm{bs}$*, il existe un arc de* x_i *vers* x_{i+1}*. On désigne* x_0 *comme la source du chemin et* x_n *comme l'image du chemin.*

On s'intéresse uniquement à la propagation des occurrences, on considère donc uniquement les chemins créés par des observations d'occurrences même si une définition similaire peut être donnée avec les observations classiques.

La séquence vide $[]$ ou une séquence singleton $[x]$ sont d'un point de vue syntaxique des chemins de propagation. Ils ne sont cependant pas définis par $\bar{\mathcal{O}}\mathrm{bs}$. Ils peuvent être utilisés pour construire un chemin bien défini par cet ensemble d'observations des occurrences en utilisant les opérateurs de jonction ou de concaténation de chemins.

Définition 25. *Opérations sur les chemins.Pour deux chemins* $P_1 : [x]$ *et* $[x] : P_2$*, on définit la jonction de ces chemins par* $(P_1 : [x]); ([x] : P_2) = P_1 : [x] : P_2$*.*

Pour deux chemins $P_1 = [x_1, x_2, \ldots, x_n]$ *et* $P_2 = [y_1, y_2, \ldots, y_n]$*, on définit la concaténation de ces chemins par* $P_1 : P_2 = [x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]$*.*

On s'intéresse aussi aux sous-chemins.

Proposition 17. *Sous-chemin. Étant donné un ensemble de relations d'observation d'occurrences* $\bar{\mathcal{O}}\mathrm{bs}$ *définissant un graphe orienté et un chemin entre deux variables de* x_0 *vers* x_n *par une séquence de variables* $[x_0, x_1, \ldots, x_n]$ *de* $\mathcal{V}\mathrm{ar}(\bar{\mathcal{O}}\mathrm{bs})$*. Alors pour tout couple* i, j *tel que* $0 \leq i < j \leq n$*, la séquence de variables* $[x_i, x_{i+1}, \ldots, x_j]$ *est un chemin bien défini par* $\bar{\mathcal{O}}\mathrm{bs}$*.*

Démonstration. Si dans le chemin initial il existe bien un arc entre chaque variable, il existe aussi dans le sous-chemin et donc ce sous-chemin est lui aussi bien défini par l'ensemble de relations d'observation d'occurrences $\bar{\mathcal{O}}\mathrm{bs}$. □

Proposition 18. *Concaténation de chemins. Étant donné un ensemble de relations d'observation d'occurrence* $\bar{\mathcal{O}}\mathrm{bs}$ *définissant un graphe orienté et deux chemins* $P_1 = [x_0, x_1, \ldots, x_k]$ *et* $P_2 = [x_{k+1}, \ldots, x_n]$*, bien définis par* $\bar{\mathcal{O}}\mathrm{bs}$*. Alors* $P_1 : P_2 = [x_0, x_1, \ldots, x_k, \ldots, x_n]$ *est un chemin bien défini par* $\bar{\mathcal{O}}\mathrm{bs}$ *s'il existe un arc du graphe orienté entre* x_k *et* x_{k+1}*.*

Proposition 19. *Jonction de chemins Étant donné un ensemble de relations d'observation d'occurrence* $\bar{\mathcal{O}}\mathrm{bs}$ *définissant un graphe orienté. Soit deux chemins* $P_1 = [x_0, x_1, \ldots, x_k]$ *et* $P_2 = [x_k, x_{k+1}, \ldots, x_n]$*. Alors* $P_1; P2 = [x_0, x_1, \ldots, x_k, \ldots, x_n]$ *est un chemin bien défini par* $\bar{\mathcal{O}}\mathrm{bs}$ *si* P_1 *et* P_2 *sont bien définis par* $\bar{\mathcal{O}}\mathrm{bs}$*.*

Démonstration. On utilise les propriétés des graphes pour montrer que dans les deux cas il existe bien un arc entre chacune des variables de la séquence défini par la concaténation ou de la jonction de deux chemins. □

Définition 26. *Chemin avec une boucle. Un chemin* P *comporte une boucle si on a :*

$$\exists P_1, P_2, P_3, \exists x : P = P_1 : x : P_2 : x : P_3$$

Un chemin comporte donc une boucle si la même variable est utilisée plusieurs fois dans la séquence de variables définissant ce chemin.

On s'intéresse aux chemins définis par une architecture et donc un ensemble d'observations d'occurrences. On ne considère cependant pas les chemins comportant des boucles.

Définition 27. *Chemins d'une architecture. Pour un ensemble d'observations d'occurrences* $\mathcal{A}\mathrm{rchi}$ *définissant l'architecture d'un système, on définit l'ensemble des chemins* $\mathcal{P}\mathrm{aths}(\mathcal{A}\mathrm{rchi})$ *de cette architecture comme l'ensemble des chemins bien définis par l'architecture et ne comportant pas de boucles.*

Le décalage logique introduit par une observation est caractérisé par l'existence d'un ensemble d'horloges d'observations donnant les différents décalages logiques possibles en chaque état. Un chemin de propagation introduit lui aussi un décalage via les observations qui le définissent. De même que pour l'observation on définit donc un ensemble d'horloges caractérisant ce décalage.

Définition 28. *Horloges d'un chemin. Soit un ensemble d'observations d'occurrences* $\bar{\mathcal{O}}\mathrm{bs}$ *et un chemin* $[x_0, x_1, \ldots, x_n]$ *bien défini par cet ensemble d'observations, soit* σ *une exécution vérifiant cet ensemble d'observation. L'ensemble des horloges qui caractérisent ce chemin pour cette exécution est notée* $\mathcal{C}([x_0, x_1, \ldots, x_n])$ *et est défini par :*

$$\mathcal{C}([x_0, x_1, \ldots, x_n]).\sigma \triangleq$$
$$\left\{ c \;\middle|\; \begin{array}{c} c \in \mathcal{C}(\mathbb{N}) \;\wedge\; \exists c_1, c_2, \ldots, c_n \in \mathcal{C}(\mathbb{N}) : \forall k \in [1..n] : \exists f_k, X_{k-1} : \\ (x_k \lessdot f_k(X_{k-1})) \in \bar{\mathcal{O}}\mathrm{bs} \;\wedge\; x_{k-1} \in X_{k-1} \;\wedge\; c_k \in \mathcal{C}(x_k \lessdot f_k(X_{k-1})).\sigma \\ \wedge\; \forall i \in \mathbb{N} : \hat{\hat{x}}_0.\sigma_{c(i)} = \hat{\hat{x}}_0.\sigma_{c_1 \circ c_2 \circ \ldots \circ c_n(i)} \end{array} \right\}$$

Justification. Dans la définition d'origine de l'horloge d'observation donnée par [Cha97], les différentes horloges d'observation définissent les possibles décalages logiques entre la source et l'image. Ces différents décalages logiques donnent, dans chaque état, un état où la valeur actuelle de l'image était prise par la source. Dans le cadre de l'observation restreinte à la modélisation de communications et de calculs et d'une observation d'occurrences $y \lessdot f(X)$, chaque horloge de l'observation donne un état où l'occurrence de X utilisée pour définir l'occurrence actuelle de y était en cours. Pour les chemins de propagation, on considère le décalage logique entre une variable x et y et pas entre X

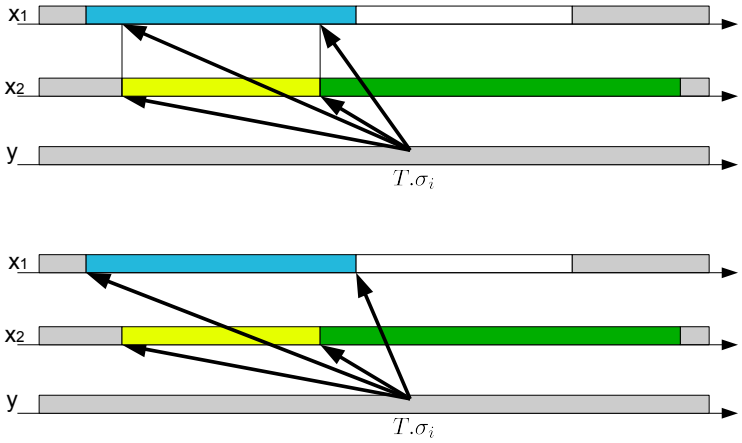

FIG. 9: Horloges d'un chemin

et y. Chaque horloge du chemin donne donc un état où l'occurrence de x utilisée pour définir l'occurrence actuelle de y était en cours et ceci indépendamment de X. Ainsi la définition de l'ensemble d'horloges d'un chemin repose sur la composition des horloges définies par chaque observation le long du chemin mais n'est pas défini uniquement par ces horloges mais par l'ensemble des horloges pointant vers la même occurrence de la source. Il s'agit de considérer tous les décalages possibles entre x et y et pas entre X et y.

On utilise un exemple donné figure 9 et permettant d'illustrer la différence. On considère une observation $y \lesssim f(x_1, x_2)$ qui définit donc deux chemins $[x_1, y]$ et $[x_2, y]$. Les horloges de l'observation donnent le décalage logique entre les valeurs du couple (x_1, x_2) et celles de y. On a une vision synchrone de ce couple. Sur la première partie de la figure 9 et pour un état σ_i les flèches donnent les possibles décalages logiques aux extrêmes, c'est-à-dire le plus ancien état pouvant être pointé et le plus récent. La valeur des horloges de l'observation dans l'état σ_i donne toutes des décalages situés entre ces deux flèches.

Lorsque l'on s'intéresse au chemin entre x_1 et y, on s'intéresse au décalage logique entre x_1 et y indépendamment des autres variables et notamment on ne veut plus être lié au couple (x_1, x_2). On veut donc que les horloges du chemin $[x_1, y]$ puissent désigner tous les états où l'occurrence de la source, qui est utilisée pour définir l'occurrence de l'image, était en cours. L'occurrence de x_1 étant plus longue que celle de x_2, sur la deuxième partie de la figure 9 il y a plus d'horloges entre x_1 et y qu'entre x_2 et y ou entre le couple (x_1, x_2) et y. C'est pour cela que la définition des horloges d'un chemin étend la composition des horloges de l'observation à toutes les horloges pointant vers la même occurrence.

On donne tout d'abord une proposition qui montre que les différentes horloges d'un chemin lient en chaque instant l'image de ce chemin à une unique occurrence de la source du chemin.

Proposition 20. *Soit un chemin d'observation P d'une variable x vers une variable y bien défini par rapport à un ensemble d'observations d'occurrences $\bar{\mathcal{O}}bs$ et une exécution σ :*

$$\forall i \in \mathbb{N} : \forall c_1, c_2 \in \mathcal{C}(P).\sigma : \hat{\hat{x}}.\sigma_{c_1(i)} = \hat{\hat{x}}.\sigma_{c_2(i)}$$

Quelque soit l'horloge du chemin, la valeur du profil temporel est la même. Le choix de l'horloge n'influence pas l'occurrence choisie mais uniquement lequel des états de cette occurrence est pointé.

Démonstration. Pour prouver cette proposition, on prouve tout d'abord que pour une observation des occurrences $y \lessdot f(X)$ et une exécution σ vérifiant cette observation, on a :

$$\forall i \in \mathbb{N} : \forall c_1, c_2 \in \mathcal{C}(y \lessdot f(X)).\sigma : \forall x \in X : \bar{x}.\sigma_{c_1(i)} = \bar{x}.\sigma_{c_2(i)}$$

Soit deux horloges, c_1, c_2 de $\mathcal{C}(y \lessdot f(X)).\sigma$ et une variable x de X. Alors pour un entier i, on a :

$\bar{y}.\sigma_i = \bar{y}.\sigma_i$
$\Rightarrow$ {définition de c_1 et c_2}
$f(\bar{X}).\sigma_{c_1(i)} = f(\bar{X}).\sigma_{c_2(i)}$
$\Rightarrow$ {définition de $f(\bar{X})$}
$\langle f(X).\sigma_{c_1(i)}, occ_{f(X)}.\sigma_{c_1(i)}\rangle = \langle f(X).\sigma_{c_2(i)}, occ_{f(X)}.\sigma_{c_2(i)}\rangle$
$\Rightarrow$ {définition d'un couple}
$occ_{f(X)}.\sigma_{c_1(i)} = occ_{f(X)}.\sigma_{c_2(i)}$
$\Rightarrow$ {définition de $occ_{f(X)}$}
$occ_x.\sigma_{c_1(i)} = occ_x.\sigma_{c_2(i)}$
$\Rightarrow$ {proposition 12}
$\bar{x}.\sigma_{c_1(i)} = \bar{x}.\sigma_{c_2(i)}$

Soit un chemin P d'une variable x à une variable y et une exécution σ pour laquelle le chemin est bien défini. En appliquant le résultat précédent à toutes les observations composant ce chemin à deux horloges c_1 et c_2 qui sont des compositions des horloges des observations du chemin, on en déduit que l'on a :

$$\forall i \in \mathbb{N} : \forall x \in X : \bar{x}.\sigma_{c_1(i)} = \bar{x}.\sigma_{c_2(i)}$$

D'après la proposition 13, on a donc :

$$\forall i \in \mathbb{N} : \forall x \in X : \hat{\bar{x}}.\sigma_{c_1(i)} = \hat{\bar{x}}.\sigma_{c_2(i)}$$

Toutes les horloges c du chemin P qui sont les composition des horloges des observation du chemin donnent donc la même valeur de $\hat{\bar{x}}.\sigma_{c(i)}$. D'après la définition des horloges $\mathcal{C}(P).\sigma$, on étend ce résultat à toutes les horloges du chemin. Ces horloges sont justement définies par rapport à l'équivalence avec une horloge c composition des horloges des observations concernant la valeur de $\hat{\bar{x}}.\sigma_{c(i)}$. □

On prouve de plus la propriété suivante :

Proposition 21. *Soit un chemin d'observation d'occurrences P d'une variable x vers une variable y bien défini par rapport à un ensemble d'observations $\bar{\mathcal{O}}bs$ et une exécution σ :*

$$\forall i, j \in \mathbb{N} : \forall c \in \mathcal{C}(P).\sigma : \bar{y}.\sigma_i = \bar{y}.\sigma_j \Rightarrow \bar{x}.\sigma_{c(i)} = \bar{x}.\sigma_{c(j)}$$

Pour un chemin bien défini, si dans deux états l'image est dans la même occurrence alors c'est la même occurrence de la source du chemin qui a été utilisée. Par contraposée, si entre deux états les occurrences de la source du chemin pointées par une horloge de cet ensemble changent, alors l'image du chemin change. Il y a donc un lien entre les occurrences de la source pointées par ces horloges et l'image du chemin. La réciproque n'est pas vraie car l'occurrence de y peut être modifiée par un changement d'occurrence d'une autre source que celle du chemin choisi.

Démonstration. On raisonne par récurrence sur la taille d'un chemin, c'est-à-dire le nombre de variables dans la séquence décrivant le chemin. Prenons un chemin de taille 2, $[x, y]$. Alors, il existe une observation $y \lessdot f(X)$ dans $\bar{\mathcal{O}}bs$ tel que $x \in X$. Pour une exécution σ et deux entiers i, j , on a alors d'après la proposition 15 :

$$
\begin{array}{ll}
 & \forall c \in \mathcal{C}(y \lessdot f(X)).\sigma : \bar{y}.\sigma_i = \bar{y}.\sigma_j \Rightarrow (\forall z \in X : \bar{z}.\sigma_{c(i)} = \bar{z}.\sigma_{c(j)}) \\
\Rightarrow & \quad \{x \in X\} \\
 & \forall c \in \mathcal{C}(y \lessdot f(X)).\sigma : \bar{y}.\sigma_i = \bar{y}.\sigma_j \Rightarrow \bar{x}.\sigma_{c(i)} = \bar{x}.\sigma_{c(j)} \\
\Rightarrow & \quad \{\text{proposition 13}\} \\
 & \forall c \in \mathcal{C}(y \lessdot f(X)).\sigma : \bar{y}.\sigma_i = \bar{y}.\sigma_j \Rightarrow \hat{\bar{x}}.\sigma_{c(i)} = \hat{\bar{x}}.\sigma_{c(j)} \\
\Rightarrow & \quad \{\text{définition des horloges d'un chemin}\} \\
 & \forall c \in \mathcal{C}([x, y]).\sigma : \bar{y}.\sigma_i = \bar{y}.\sigma_j \Rightarrow \hat{\bar{x}}.\sigma_{c(i)} = \hat{\bar{x}}.\sigma_{c(j)} \\
\Rightarrow & \quad \{\text{proposition 13}\} \\
 & \forall c \in \mathcal{C}([x, y]).\sigma : \bar{y}.\sigma_i = \bar{y}.\sigma_j \Rightarrow \bar{x}.\sigma_{c(i)} = \bar{x}.\sigma_{c(j)}
\end{array}
$$

Donc la propriété est vraie pour un chemin de taille 2. Supposons qu'elle soit vraie pour un chemin de taille n. Alors soit un chemin $[x_0, x_1, \ldots, x_n]$ de taille $n+1$. Les deux chemins $[x_0, x_1]$ et $[x_1, \ldots, x_n]$ sont deux chemins de taille 2 et n. Donc par hypothèse de récurrence on a les deux propriétés suivantes pour tous entiers i, j :

$$\forall c \in \mathcal{C}([x_0, x_1]).\sigma : \bar{x_1}.\sigma_i = \bar{x_1}.\sigma_j \Rightarrow \bar{x_0}.\sigma_{c(i)} = \bar{x_0}.\sigma_{c(j)}$$

$$\forall c \in \mathcal{C}([x_1, \ldots, x_n]).\sigma : \bar{x_n}.\sigma_i = \bar{x_n}.\sigma_j \Rightarrow \bar{x_1}.\sigma_{c(i)} = \bar{x_1}.\sigma_{c(j)}$$

Soit une horloge c du chemin $[x_0, x_1, \ldots, x_n]$. Alors :

$$
\begin{array}{l}
\exists c_1, c_2, \ldots, c_n : \forall k \in [1..n], \exists f_k, X_{k-1} : (x_k \lessdot f_k(X_{k-1})) \in \bar{\mathcal{O}}bs \ \wedge \ x_{k-1} \in X_{k-1} \ \wedge \\
\forall i \in \mathbb{N} : x_k.\sigma_i = f_k(X_{k-1}.\sigma_{c_k(i)}) \ \wedge \ \hat{\bar{x}}_0.\sigma_{c_1'(i)} = \hat{\bar{x}}_0.\sigma_{c_1(i)} \ \wedge \\
\hat{\bar{x}}_0.\sigma_{c(i))} = \hat{\bar{x}}_0.\sigma_{c_1 \circ c_2 \circ \ldots \circ c_n(i)}
\end{array}
$$

L'horloge c_1 est une horloge du chemin $[x_0, x_1]$ et l'horloge $c_2' = c_2 \circ \ldots \circ c_n$ est une horloge du chemin $[x_1, \ldots, x_n]$ d'après la définition des horloges d'un chemin. En appliquant l'hypothèse de récurrence à ces horloges, on a :

$$
\begin{array}{ll}
 & \forall i, j \in \mathbb{N} : \bar{x_n}.\sigma_i = \bar{x_n}.\sigma_j \Rightarrow \bar{x_0}.\sigma_{c_1 \circ c_2'(i)} = \bar{x_0}.\sigma_{c_1 \circ c_2'(j)} \\
\Rightarrow & \quad \{\text{proposition 13}\} \\
 & \forall i, j \in \mathbb{N} : \hat{\bar{x}}_n.\sigma_i = \hat{\bar{x}}_n.\sigma_j \Rightarrow \hat{\bar{x}}_0.\sigma_{c_1 \circ c_2'(i)} = \hat{\bar{x}}_0.\sigma_{c_1 \circ c_2'(j)} \\
\Rightarrow & \quad \{\text{définition de } c_1 \text{ et } c_2'\} \\
 & \forall i, j \in \mathbb{N} : \hat{\bar{x}}_n.\sigma_i = \hat{\bar{x}}_n.\sigma_j \Rightarrow \hat{\bar{x}}_0.\sigma_{c(i)} = \hat{\bar{x}}_0.\sigma_{c(j)}
\end{array}
$$

On généralise le résultat à tout horloge du chemin en utilisant la proposition 20. Par récurrence, la propriété est donc vraie quelque soit la taille du chemin. □

On donne la proposition suivante :

Proposition 22. *Soit un ensemble d'observations d'occurrences* $\bar{\mathcal{O}}bs$*, une variable* x *et deux chemins* $P_1 : [x]$ *et* $[x] : P_2$ *bien définis par cet ensemble d'observations. Soit* σ *une exécution vérifiant cet ensemble d'observation, on a alors :*

$$\forall c_1 \in \mathcal{C}(P_1 : [x]).\sigma, \forall c_2 \in \mathcal{C}([x] : P_2).\sigma : c_1 \circ c_2 \in \mathcal{C}(P_1 : [x] : P_2).\sigma$$

Cette proposition permet de construire des horloges d'un chemin à partir de celles des sous-chemins. Il n'est cependant pas nécessairement possible de construire l'intégralité des horloges d'un chemin à partir de celles des sous-chemins.

Démonstration. Soit un chemin $[x_0, x_1, \ldots, x_n]$, deux sous-chemins $[x_0, x_1, \ldots, x_l]$ et $[x_l, \ldots, x_n]$ et deux horloges c', c'' tel que :

$$c' \in \mathcal{C}([x_0, x_1, \ldots, x_l]).\sigma \wedge c'' \in \mathcal{C}([x_l, \ldots, x_n]).\sigma$$

Soit un entier i. D'après la définition des horloges d'un chemin, on a :

$$\begin{array}{l}\exists c_{l+1}, c_{l+2}, \ldots, c_n : \forall k \in [l+1..n], \exists f_k, X_{k-1} : (x_k \trianglelefteq f_k(X_{k-1})) \in \bar{\mathcal{O}}bs \\ \wedge\ x_{k-1} \in X_{k-1} \ \wedge\ c' \in \mathcal{C}(\mathbb{N}) \wedge \\ x_k.\sigma_i = f_k(X_{k-1}.\sigma_{c_k(i)}) \ \wedge\ \hat{\hat{x}}_l.\sigma_{c''(i)} = \hat{\hat{x}}_l.\sigma_{c_{l+1} \circ c_{l+2} \circ \ldots \circ c_n(i)}\end{array}$$

et

$$\begin{array}{l}\exists c_1, c_2, \ldots, c_l : \forall k \in [1..l], \exists f_k, X_{k-1} : (x_k \trianglelefteq f_k(X_{k-1})) \in \bar{\mathcal{O}}bs \\ \wedge\ x_{k-1} \in X_{k-1} \ \wedge\ c' \in \mathcal{C}(\mathbb{N}) \wedge \\ x_k.\sigma_i = f_k(X_{k-1}.\sigma_{c_k(i)}) \ \wedge\ \hat{\hat{x}}_0.\sigma_{c'(c''(i))} = \hat{\hat{x}}_0.\sigma_{c_1 \circ c_2 \circ \ldots \circ c_l(c''(i))}\end{array}$$

L'horloge $c_1 \circ c_2 \circ \ldots \circ c_l$ appartient elle aussi à $\mathcal{C}([x_1, \ldots, x_l]).\sigma$, étant la composition des horloges d'observations et d'après la définition des horloges d'un chemin. Or on a $\hat{\hat{x}}_l.\sigma_{c''(i)} = \hat{\hat{x}}_l.\sigma_{c_{l+1} \circ c_{l+2} \circ \ldots \circ c_n(i)}$ et donc d'après les propositions 13 et 21 pour le chemin $[x_1, \ldots, x_l]$ on a :

$$\hat{\hat{x}}_0.\sigma_{c_1 \circ c_2 \circ \ldots \circ c_l(c''(i))} = \hat{\hat{x}}_0.\sigma_{c_1 \circ c_2 \circ \ldots \circ c_l(c_{l+1} \circ c_{l+2} \circ \ldots \circ c_n(i))} = \hat{\hat{x}}_0.\sigma_{c'(c''(i))}$$

On en déduit donc que l'on a :

$$\begin{array}{l}\exists c_1, c_2, \ldots, c_n : \forall k \in [1..n], \exists f_k, X_{k-1} : (x_k \trianglelefteq f_k(X_{k-1})) \in \bar{\mathcal{O}}bs \\ \wedge\ x_{k-1} \in X_{k-1} \ \wedge\ c_k \in \mathcal{C}(\mathbb{N}) \wedge \\ x_k.\sigma_i = f_k(X_{k-1}.\sigma_{c_k(i)}) \ \wedge\ \hat{\hat{x}}_0.\sigma_{c' \circ c''(i)} = \hat{\hat{x}}_0.\sigma_{c_1 \circ c_2 \circ \ldots \circ c_n(i)}\end{array}$$

et donc $c' \circ c'' \in \mathcal{C}([x_0, x_1, \ldots, x_n]).\sigma$. □

3.4 RÉCAPITULATIF

Dans ce chapitre, nous avons introduit les outils qui nous permettent par la suite de donner une spécification d'un système temps réel centrée sur les propriétés temporelles des valeurs des variables et leur propagation dans le système. Nous avons premièrement introduit le temps comme une variable T qui mesure ce temps en chaque état et prend ses valeurs dans $\mathbb{N}$. Afin de décrire le comportement temporel des variables du système et donc l'évolution des valeurs prises par ces variables au cours du temps, le profil temporel a été défini puis a ensuite été adapté à la notion d'occurrence.

Pour décrire la propagation des valeurs dans le système, on utilise la relation d'observation. Cette relation permet de modéliser les communications et les calculs effectués par le système. L'horloge d'une observation donne de plus une mesure du décalage logique introduit par une relation d'observation. Un ensemble de relations d'observation définit une architecture. On définit alors des chemins de propagation caractérisant la propagation des valeurs d'une variable vers une autre variable.

Ces définitions peuvent être données sur les valeurs ou sur les occurrences des variables. On précise quelles sont les définitions qui sont utilisées dans la suite du manuscrit en simplifiant le vocabulaire utilisé :

- le profil temporel d'une variable désigne par la suite le profil temporel des occurrences de cette variable ;

- l'instant de mise à jour d'une variable est donc la valeur du profil temporel en l'état considéré ;
- la stabilité d'une variable désigne la stabilité des occurrences de cette variable ;
- la durée entre deux mises à jour est donc la valeur de la variable stabilité en l'état considéré ;
- les observations considérées par la suite sont les observations des occurrences ;
- les chemins de propagation sont uniquement construits par un ensemble d'observations des occurrences.

Par la suite, on se réfère donc uniquement aux définitions données dans le cadre des occurrences.

Dans le chapitre suivant, on utilise ces définitions et propriétés pour définir les propriétés temps réel d'un système.

4

SPÉCIFICATION D'UN SYSTÈME PAR L'OBSERVATION

Nous avons vu dans le chapitre précédent comment modéliser l'architecture d'un système et décrire les chemins de propagation dans ce système. Nous avons aussi introduit le profil temporel et les occurrences permettant de décrire le comportement d'une variable au cours d'une exécution. Nous n'avons cependant pas donné de propriétés temporelles sur ces chemins de propagation et sur le comportement des variables. Dans ce chapitre, on définit un ensemble de propriétés temporelles permettant de spécifier le comportement temps réel d'un système. Il s'agit d'utiliser le profil temporel d'une variable pour contraindre le comportement temporel d'une variable et son rythme de mise à jour. Puis, on définit des propriétés contraignant la propagation des occurrences des variables dans le système. Ces propriétés se définissent par rapport aux chemins de propagation et aux décalages logiques introduits par ces chemins.

Une spécification d'un système est alors donnée par un couple architecture, ensemble de propriétés temps réel. On donne alors les différentes propriétés que peut avoir une telle spécification ainsi que les liens entre les différentes propriétés temps réel du système.

4.1 COMPORTEMENT TEMPOREL D'UNE VARIABLE

On veut caractériser le rythme de mise à jour d'une variable et donc les valeurs du profil temporel et de la stabilité de cette variable. Il s'agit notamment de limiter le temps écoulé entre deux mise à jours et d'assurer ainsi une évolution régulière du système. On définit tout d'abord une propriété bornant le temps écoulé entre deux mises à jour.

Définition 29. *Sporadicité. Soit une exécution séparée* σ *et une variable* x*. Le comportement de* x *lors de cette exécution est sporadique de paramètres* δ, Δ *si on a :*

$$\sigma \vDash x\ \{\mathrm{Sporadic}(\delta, \Delta)\} \triangleq \forall i \in \mathbb{N} : d_{\hat{x}}.\sigma_i \in [\delta..\Delta]$$

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$*.*

En contraignant de cette manière les valeurs de la stabilité d'une variable, on indique que le temps entre deux mises à jour et donc la durée d'une occurrence est au minimum de δ et au maximum de Δ. S'il n'y a pas de temps minimum entre deux mises à jour on utilise 0 pour δ et s'il n'y a pas de maximum, on utilise $+\infty$ pour Δ. Il s'agit de spécifier la vitesse de mise à jour qu'une variable doit avoir pour que ses valeurs restent correctes, par exemple de donner une borne supérieure pour qu'une entrée du système soit mise à jour suffisamment souvent pour suivre l'évolution de l'environnement. Une borne inférieure peut traduire un temps minimum pour obtenir une mesure précise.

Pour certaines variables, on donne une définition d'un comportement plus précis.

Définition 30. *Périodicité. Soit une exécution séparée* σ *et une variable* x*. Le comportement de* x *lors de cette exécution est périodique de période* P *et de gigue* J *si on a :*

$$\sigma \vDash x\ \{\mathrm{Periodic}(P, J)\} \triangleq \exists \delta = \langle \delta_i | i \in \mathbb{N}^* \wedge \delta_i \in [-J..J] \wedge \delta_0 = 0 \rangle : \{\hat{\hat{x}}.\sigma_i | i \in \mathbb{N}\} = \{n * P + \delta_n | n \in \mathbb{N}\}$$

Cette définition dit que pour une exécution où la variable x est périodique, les instants de mise à jour de x sont situés autour des instants $n * T$ à la gigue près. De plus pour chacun de ces instants et donc à chaque période il y a une mise à jour. La périodicité est une contrainte sur le profil temporel plus forte que la sporadicité.

Proposition 23. *Lien entre périodicité et sporadicité. Soit une exécution séparée σ et une variable x. Alors pour deux entiers P et J, on a :*

$$\sigma \vDash x\ \{\mathtt{Periodic}(P,J)\} \Rightarrow \sigma \vDash x\ \{\mathtt{Sporadic}(max(0,P-2J),P+2J)\}$$

Démonstration. Soit un entier i et les deux instants de mises à jour successifs $\hat{\hat{x}}.\sigma_i, \hat{\hat{x}}.\sigma_j$ tel que $\hat{\hat{x}}.\sigma_i < \hat{\hat{x}}.\sigma_j$, alors :

$$\begin{array}{ll} & \exists n \in \mathbb{N}, \exists \delta_i, \delta_j \in [-J..J] : \hat{\hat{x}}.\sigma_i = n * P + \delta_i \ \wedge\ \hat{\hat{x}}.\sigma_j = (n+1) * P + \delta_j \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & \exists \delta_i, \delta_j \in [-J..J] : \hat{\hat{x}}.\sigma_j - \hat{\hat{x}}.\sigma_i = P + \delta_j - \delta_i \\ \Rightarrow & \quad \{\text{définition de } \delta_i \text{ et } \delta_j\} \\ & \hat{\hat{x}}.\sigma_j - \hat{\hat{x}}.\sigma_i \in [P-2J..P+2J] \end{array}$$

Or $\hat{\hat{x}}$ est constant entre σ_i et σ_{j-1} donc x est constant entre ces deux valeurs. Donc $d_{\bar{x}}.\sigma_i \in [(P-2J)..(P+2J)]$. Ceci est aussi vrai pour δ_0 car cette valeur est bien entre $[-J..J]$. On a donc $x\ \{\mathtt{Sporadic}(P-2J,P+2J)\}$. □

Proposition 24. *Vivacité. Soit une exécution séparée et non Zénon σ et une variable x. Alors s'il existe $\Delta \in \mathbb{N}$ tel que $x\ \{\mathtt{Sporadic}(0,\Delta)\}$ alors la variable x est vivace pour l'exécution σ.*

Démonstration. Soit $i \in \mathbb{N}$ alors, comme le système est non Zénon, $\exists j : T.\sigma_j \geq T.\sigma_i + \Delta$. Donc d'après la définition de $d_{\bar{x}} : \exists k \in [i..j] : x.\sigma_k \neq x.\sigma_i$ donc x est vivace. □

4.2 PROPRIÉTÉS SUR LA PROPAGATION DES OCCURRENCES

4.2.1 *Forme générale*

Une variable image d'une observation est mise à jour avec de nouvelles occurrences des sources de l'observation. A travers ces occurrences, l'image est alors liée à de nouvelles occurrences des chemins dont elle est l'image. On introduit des propriétés qui définissent la validité temporelle des occurrences des sources des chemins. Une mise à jour de l'image d'un chemin est alors possible uniquement si elle est effectuée avec une occurrence valide de la source de chaque chemin.

On donne les critères de validité comme un ensemble de prédicats sur un chemin de propagation devant être vérifié en tout état d'une exécution.

Définition 31. *Propriétés temps réel d'un chemin. Étant donné un chemin P entre deux variables y et x, ce chemin est paramétré par un ensemble de prédicats $\{\mathtt{Predicate}_1(\delta_1,\Delta_1),\ldots,\mathtt{Predicate}_n(\delta_n,\Delta_n)\}$ si on a :*

$$\begin{array}{l} \sigma \vDash P\ \{\mathtt{Predicate}_1(\delta_1,\Delta_1),\ldots,\mathtt{Predicate}_n(\delta_n,\Delta_n)\} \triangleq \\ \exists c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : \left(\begin{array}{c} \mathtt{Predicate}_1(P,c,\delta_1,\Delta_1).\sigma_i \\ \wedge\ \ldots\ \wedge \\ \mathtt{Predicate}_n(P,c,\delta_n,\Delta_n).\sigma_i \end{array} \right) \end{array}$$

et où

$$\forall i \in [1..n] : \mathtt{Predicate}_i \in \{\mathtt{Latency},\mathtt{Lag},\mathtt{Shift},\mathtt{Freshness},\mathtt{Fitness},\mathtt{Stability}\}$$

Dans cette définition, on utilise une horloge du chemin. En un état, l'utilisation d'une telle horloge permet notamment de se référer à un état où est assignée à la source l'occurrence qui est actuellement utilisée pour définir l'occurrence de l'image. Alors, le comportement d'un chemin est valide lorsqu'en tout état, la valeur de la source utilisée pour définir la valeur de l'image respecte un ensemble de prédicats.

La sémantique d'une telle propriété dépend des prédicats utilisés, prédicats appartenant à un ensemble de prédicats prédéfinis dont on donne ci-dessous la définition.

4.2.2 *Propriétés temps réel d'un chemin*

Ces prédicats sont définis grâce au décalage logique introduit par un chemin et aux différentes variables temporelles introduites précédemment, notamment le profil temporel de la source et de l'image. Ils sont construits en donnant des bornes sur la différence entre deux instants caractérisant le comportement temporel de la source et de l'image. Ces instants sont définis en utilisant le profil temporel de la source et de l'image, la variable T et l'horloge du chemin.

Les définitions de chacun des prédicats sont illustrées par une figure représentant un chemin de propagation d'une variable x vers une variable y. Sur chacune des figures, le temps s'écoule de gauche à droite, une ligne représente l'évolution des occurrences prises par une variable au cours du temps et chaque zone de couleur correspond à une occurrence de cette variable. Si dans un état l'occurrence de y est représentée de couleur blanche c'est qu'elle est construite à partir de l'occurrence de l'image représentée de la même couleur blanche.

Définition 32. *Latence. Pour une exécution σ, le prédicat de latence pour un état σ_i et pour un chemin P d'une variable x vers une variable y est défini par :*

$$\mathtt{Latency}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq T.\sigma_i - \hat{x}.\sigma_{c(i)} < \Delta$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$.

La latence borne le temps écoulé entre l'instant actuel et le moment où l'occurrence de x utilisée pour définir la valeur actuelle de y a été mise à jour. Il borne en tout instant le temps écoulé depuis l'apparition de l'occurrence de la source dans le système pour qu'elle puisse être utilisée pour définir une occurrence de l'image.

Le prédicat est illustré figure 10. La flèche indique le décalage entre l'instant de mise à jour de l'occurrence de x et l'instant actuel. Elle illustre donc le temps écoulé depuis l'apparition de cette occurrence de x dans le système. Pour la variable y, cette occurrence de x peut uniquement être utilisée tant que cette pente est située entre les lignes pointillées. Ces lignes pointillées donnent les bornes des pentes possibles pour cette flèche d'après le prédicat de latence.

Supposons qu'il existe un chemin d'une variable x vers une variable y et que l'on veuille que ce chemin respecte une latence $\mathtt{Latency}(2, 5)$. Une occurrence de x apparue à $T = 10$ ne pourra pas être liée à une occurrence de y avant que 2 unités de temps se soient écoulées et donc à partir de la 12-ème unité de temps. Ensuite, elle ne pourra plus être liée à y à partir de la 15-ème unité de temps. La variable y doit donc être mise à jour et liée à une occurrence de x plus récente au plus tard à $T = 15$.

Définition 33. *Lag. Pour une exécution σ, le prédicat de lag pour un état σ_i et pour un chemin P d'une variable x vers une variable y est défini par :*

$$\mathtt{Lag}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq \hat{y}.\sigma_i - \hat{x}.\sigma_{c(i)} < \Delta$$

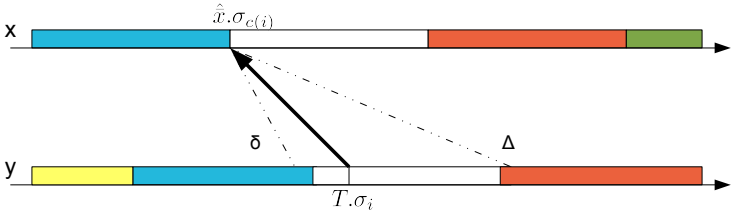

FIG. 10: Le prédicat de latence

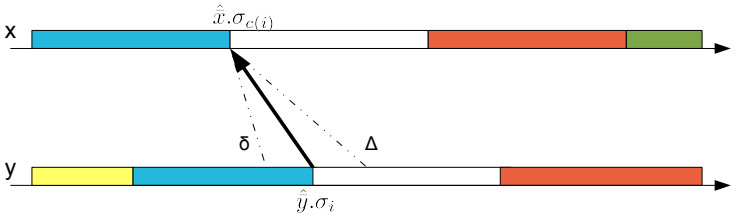

FIG. 11: Le prédicat de lag

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$ *et* $\delta < \Delta$.

Ce prédicat est illustré figure 11. La flèche représente le temps écoulé entre le moment d'apparition d'une nouvelle occurrence de x et le moment d'apparition d'une nouvelle occurrence de y issue de cette occurrence de la source. Il s'agit en fait du temps écoulé pour que la mise à jour de x se répercute en une mise à jour de y. Le lag borne ce temps, les lignes pointillées représentent les différentes pentes possibles et indiquent les instants où une mise à jour de y peut se faire avec cette occurrence de x.

Supposons qu'il existe un chemin d'une variable x vers une variable y et que l'on veuille que ce chemin respecte un lag $\mathrm{Lag}(3, 8)$. Une occurrence de x apparue à un instant à $T = 10$ peut être utilisée pour mettre à jour y uniquement à un instant entre 13 et 18. Par rapport à la latence, le lag indique uniquement quand une mise à jour avec une occurrence est possible mais ne dit pas combien de temps une occurrence peut être conservée et quand devra avoir lieu la prochaine mise à jour.

Définition 34. *Décalage. Pour une exécution* σ*, le prédicat de décalage pour un état* σ_i *et pour un chemin* P *d'une variable* x *vers une variable* y *est défini par :*

$$\mathrm{Shift}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq T.\sigma_i - T.\sigma_{c(i)} < \Delta$$

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$ *et* $\delta < \Delta$.

L'horloge d'observation et la différence $i - c(i)$ représente le décalage logique introduit le long d'un chemin de propagation. Le prédicat de décalage borne le décalage temporel associé à ce décalage logique. Il s'agit alors d'une borne sur le temps écoulé entre l'instant actuel et un instant où l'occurrence de x utilisée pour définir la valeur actuelle de y était affectée à x.

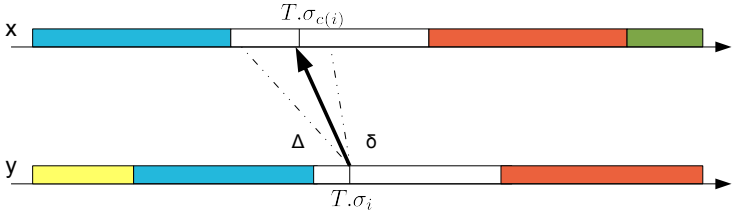

FIG. 12: Le prédicat de décalage

Sur la figure 12, la flèche donne l'instant pointé par l'horloge du chemin de propagation. La pente de cette flèche doit être située entre les lignes pointillées représentant les bornes du prédicat de décalage. Supposons que l'on ait un prédicat de décalage $\mathtt{Shift}(1,4)$. Dans ce cas, en tout instant, pour une occurrence de l'image, il faut qu'il existe un état entre 1 et 4 unités de temps avant et où l'occurrence de la source actuellement observée était celle de la source. A cet instant, comme cette occurrence était prise par la source, elle était alors valide.

C'est une propriété abstraite étant donné que sa définition est indépendante des événements de mise à jour. De plus il y a plusieurs instants pendant lesquels l'occurrence utilisée pour mettre à jour l'image était affectée à la source et donc plusieurs décalages logiques possibles, représentés par plusieurs horloges d'observation. Il suffit simplement qu'une des horloges satisfasse le prédicat.

Définition 35. *Fraîcheur. Pour une exécution* σ*, le prédicat de fraîcheur pour un état* σ_i *et pour un chemin* P *d'une variable* x *vers une variable* y *est défini par :*

$$\mathtt{Freshness}(P,c,\delta,\Delta).\sigma_i \triangleq \delta \leq T.\sigma_{c(i)} - \hat{x}.\sigma_{c(i)} < \Delta$$

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$ *et* $\delta < \Delta$.

Cette propriété restreint les instants que peut pointer l'horloge du chemin aux instants où une occurrence de x est suffisamment fraîche (mais pas trop) en bornant le temps écoulé entre l'apparition de cette occurrence et l'état pointé par l'horloge du chemin. Sur la figure 13, et comme pour la figure 12, la flèche donne l'instant pointé par l'horloge du chemin de propagation. On ne s'intéresse pas à la pente de cette flèche mais aux propriétés de l'instant pointé. Pour une propriété de fraîcheur $\mathtt{Freshness}(2,4)$, il faut que cette flèche pointe vers un instant situé entre 2 et 4 unités de temps après la mise à jour de la source correspondant à l'occurrence observée. On s'intéresse donc au temps écoulé depuis l'apparition de l'occurrence observée et il faut qu'il existe un état où cette occurrence aurait pu être lue en étant suffisamment fraîche.

Étant donné que l'on considère l'état pointé par l'horloge du chemin, la propriété de décalage et la propriété de fraîcheur dépendent l'une de l'autre. Il doit exister une horloge de l'observation satisfaisant les deux prédicats. On veut par exemple par la propriété de décalage que l'horloge d'observation pointe vers des états récents et où de plus l'occurrence de la source est fraîche. Si l'on donne une borne supérieure au décalage et à la fraîcheur alors il doit exister un décalage entre l'instant actuel et un instant où l'occurrence de la source était fraîche, décalage qui est limité par la borne supérieure sur la fraîcheur. Cela implique alors que le décalage entre la source et l'image soit faible et qu'en plus l'on observe des occurrences récemment mises à jour.

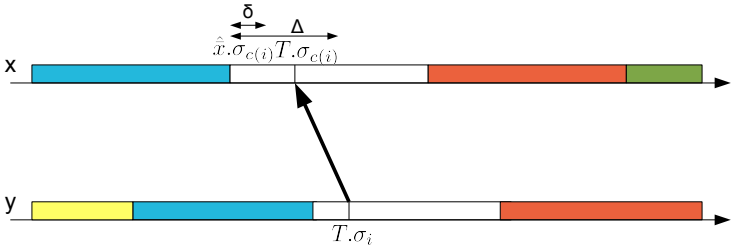

FIG. 13: Le prédicat de fraîcheur

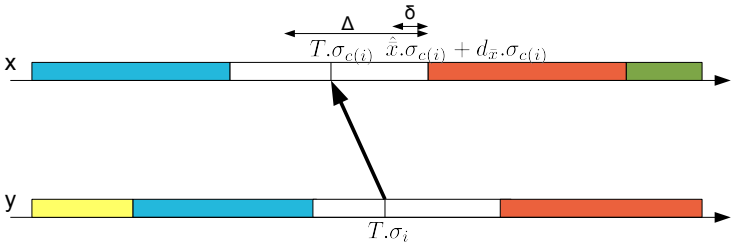

FIG. 14: Le prédicat de pérennité

Définition 36. *Pérennité. Pour une exécution* σ*, le prédicat de pérennité pour un état* σ_i *et pour un chemin* P *d'une variable* x *vers une variable* y *est défini par :*

$$\mathtt{Fitness}(P, c, \delta, \Delta).\sigma_i \triangleq \delta < \hat{x}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - T.\sigma_{c(i)} \leq \Delta$$

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$ *et* $\delta < \Delta$.

Cette propriété est semblable à la fraîcheur et est illustrée de manière semblable figure 14. Cependant la restriction des états visés par les horloges du chemin est définie par rapport à la proximité d'une prochaine mise à jour de x dans l'état pointé par l'horloge. On valide ou pas cette horloge selon le temps qui s'écoulera jusqu'à la prochaine mise à jour et donc la fin de cette occurrence. Pour une propriété de pérennité $\mathtt{Fitness}(3, 10)$, l'horloge de l'observation doit alors pointer vers un état situé entre 3 et 10 unités de temps avant la fin de l'occurrence. Si les horloges vérifiant le prédicat de décalage pointent uniquement vers des états à moins de trois unités de temps de la prochaine mise à jour et donc de la disparition de l'occurrence, la conjonction du décalage et de la pérennité n'est pas possible. On considère donc que cette occurrence n'est pas valide.

On remarque que l'intervalle défini pour cette différence est ouvert au début et fermé à la fin contrairement aux autres propriétés. Il s'agit tout simplement que l'intervalle soit fermé au début et ouvert à la fin pour les valeurs du temps T. Ainsi comme pour les autres propriétés, lorsque l'on parcourt les états satisfaisant la propriété dans l'ordre chronologique, on décrit un intervalle commençant par un fermé et finissant par un ouvert.

Définition 37. *Stabilité. Pour une exécution* σ*, le prédicat de stabilité pour un état* σ_i *et pour un chemin* P *d'une variable* x *vers une variable* y *est défini par :*

$$\mathtt{Stability}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq d_{\bar{x}}.\sigma_{c(i)} < \Delta$$

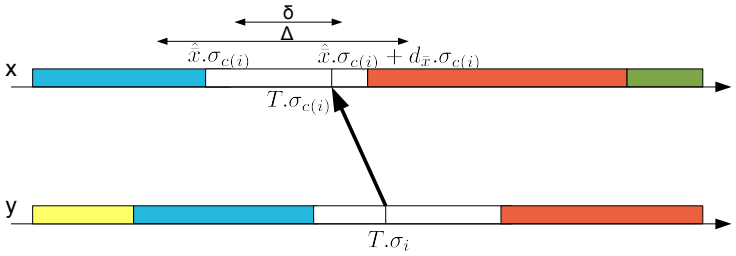

FIG. 15: Le prédicat de stabilité

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$ *et* $\delta < \Delta$.

On considère la durée pendant laquelle l'occurrence de x dure et donc la valeur de la variable stabilité. Pour un prédicat de stabilité $\texttt{Stability}(4, 8)$, on pourra uniquement lier les occurrences de l'image à des occurrences que la source conserve au minimum 4 unités de temps et au maximum 8 unités de temps. On élimine donc les occurrences conservées un temps trop faible ou au contraire conservées un temps trop important. On peut par exemple considérer qu'une occurrence conservée moins de 4 unités de temps est un phénomène transitoire inintéressant. On peut au contraire s'intéresser uniquement aux occurrences conservées un faible instant. Ce prédicat est illustré figure 15. La flèche représente l'état observé, il correspond bien à une occurrence dont la durée est comprise entre δ et Δ.

4.2.3 *Différences et complémentarité des différents propriétés*

On distingue, parmi le groupe de prédicats, plusieurs types de propriétés différentes. La propriété de latence borne le décalage entre l'instant actuel et la date de mise à jour de l'occurrence de la source. Elle détermine donc quand une occurrence de l'image peut être liée à cette occurrence de la source. Lorsque l'on est dans un instant ne vérifiant pas ces conditions, il faut utiliser une autre occurrence de la source. Notamment une mise à jour de l'image avec une nouvelle occurrence de la source doit être faite avant que la borne maximum soit atteinte .

La propriété de lag est basée sur les événements et indique juste quand une mise à jour de l'image peut être faite avec une occurrence de la source d'un chemin. Elle n'impose cependant aucune condition sur les instants où cette occurrence de l'image peut être conservée. Cette propriété n'implique donc pas de faire de nouvelles mises à jour de l'image avec de nouvelles occurrences de la source. De par sa définition sur les événements, cette propriété peut être utilisée pour donner des propriétés sur les mécanismes utilisées. En donnant une borne minimum, on donne le temps minimum pour lire la valeur de la source puis effectuer le calcul ou la communication impliquant la mise à jour de l'image.

Les propriétés de fraîcheur, de pérennité et de stabilité indiquent juste si une occurrence de la source peut être utilisée mais n'imposent rien sur les instants où elle peut être utilisée. En effet, elles portent uniquement sur les propriétés de l'occurrence source. A l'inverse, la propriété de décalage ne porte pas sur les propriétés de l'occurrence source. Cependant, la combinaison de la fraîcheur ou de la pérennité avec le décalage fait que le décalage entre un état valide de l'occurrence source et l'instant courant est

borné. Dans le cas d'une borne supérieure sur le décalage, on peut être amené à devoir mettre à jour l'image avec une nouvelle occurrence de la source lorsque le décalage avec les états satisfaisants de la source devient trop important.

On a donc des propriétés qui imposent des conditions sur les occurrences de la source que l'on peut utiliser et des propriétés qui donnent les instants où l'on peut utiliser une occurrence valide. La propriété de décalage permet de faire la jonction entre ces deux types de propriétés. La propriété de lag donne, quant à elle, les instants où l'on peut effectuer une mise à jour.

4.2.4 *Quelques résultats*

On donne deux propriétés simples sur un chemin paramétré par un ensemble de prédicats.

Proposition 25. *Étant donné un chemin* P *entre deux variables* y *et* x*, chemin paramétré par un ensemble de prédicats* $\{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\}$*, alors on a :*

$$\begin{array}{l}\sigma \vDash P\ \{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\} \Rightarrow \\ \forall K \subset [1..n] : \sigma \vDash P\ \{\mathtt{Predicate}_k(\delta_k, \Delta_k) | k \in K\}\end{array}$$

Donc si un chemin vérifie un ensemble de prédicats alors il vérifie tout sous-ensemble de ces prédicats.

Démonstration. La preuve est naturelle à partir des propriétés de la conjonction sachant que $(A \wedge B) \Rightarrow A$. □

Proposition 26. *Étant donné un chemin* P *entre deux variables* y *et* x*, chemin paramétré par un ensemble de prédicats* $\{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\}$*, alors on a :*

$$\begin{array}{l}\sigma \vDash P\ \{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\} \Rightarrow \\ \left(\begin{array}{l} \forall k \in [1..n] : \forall \delta'_k \in \mathbb{N} : \delta'_k \leq \delta_k, \forall \Delta'_K \in \mathbb{N} \cup \{+\infty\} : \Delta'_k \geq \Delta_k : \\ \quad \sigma \vDash P\ \{\mathtt{Predicate}_1(\delta'_1, \Delta'_1), \ldots, \mathtt{Predicate}_n(\delta'_n, \Delta'_n)\} \end{array}\right)\end{array}$$

Démonstration. La preuve vient du fait que chaque prédicat correspond à une inégalité. Si une inégalité est respectée alors toute inégalité avec des bornes plus larges est elle aussi respectée. □

On définit la forme normale d'une propriété sur un chemin de propagation.

Définition 38. *Forme normale d'une propriété sur un chemin. Étant donné un chemin* P *entre deux variables* y *et* x *et une propriété* prop *sur ce chemin. Cette propriété est sous la forme normale si :*

$$\begin{array}{l}\exists \delta_{Lat}, \delta_{Lag}, \delta_{Shi}, \delta_{Fre}, \delta_{Fit}, \delta_{Sta} \in \mathbb{N}, \\ \exists \Delta_{Lat}, \Delta_{Lag}, \Delta_{Shi}, \Delta_{Fre}, \Delta_{Fit}, \Delta_{Sta} \in \mathbb{N} \cup \{+\infty\} : \\ \mathtt{prop} = P \left\{ \begin{array}{l} \mathtt{Latency}(\delta_{Lat}, \Delta_{Lat}), \mathtt{Lag}(\delta_{Lag}, \Delta_{Lag}), \mathtt{Shift}(\delta_{Shi}, \Delta_{Shi}), \\ \mathtt{Freshness}(\delta_{Fre}, \Delta_{Fre}), \mathtt{Fitness}(\delta_{Fit}, \Delta_{Fit}), \mathtt{Stability}(\delta_{Sta}, \Delta_{Sta}) \end{array} \right\}\end{array}$$

Cette forme normale est purement syntaxique et ne s'intéresse pas aux valeurs des paramètres de chaque prédicat. Pour pouvoir exprimer les propriétés d'un chemin sous une forme normale, on s'appuie sur la propriété suivante :

Proposition 27. *Chronologie des variables temporelles. Étant donné un chemin* P *d'une variable* x *vers une variable* y *et une exécution* σ *alors on a :*

$$\sigma \vDash P \Rightarrow \sigma \vDash \forall c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : \hat{\bar{x}}.\sigma_{c(i)} \leq T.\sigma_i \ \wedge \ \hat{\bar{x}}.\sigma_{c(i)} \leq \hat{\bar{y}}.\sigma_i$$
$$\wedge\ T.\sigma_{c(i)} \leq T.\sigma_i \ \wedge \ \hat{\bar{x}}.\sigma_{c(i)} \leq T.\sigma_{c(i)} \ \wedge \ T.\sigma_i < \hat{\bar{x}}.\sigma_i + d_{\bar{x}}.\sigma_i \ \wedge \ T.\sigma_{c(i)} < \hat{\bar{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)}$$

Démonstration. Soit $c \in \mathcal{C}(P).\sigma$. Premièrement, c est une horloge et donc on a $c(i) \leq i$ donc comme la variable T est croissante, on a bien $\underline{T.\sigma_{c(i)} \leq T.\sigma_i}$.

Puis on reprend la définition de $\hat{\bar{x}}$.

$$\forall i \in \mathbb{N} : \hat{\bar{x}}.\sigma_i \triangleq T.\sigma_{min\{j|\forall k \in [j..i]:\ \bar{x}.\sigma_i = \bar{x}.\sigma_k\}}$$

Par définition on a $min\{j|\forall k \in [j..i] :\ \bar{x}.\sigma_i = \bar{x}.\sigma_k\} \leq i$ donc comme T est croissante, $T.\sigma_{min\{j|\forall k \in [j..i]:\ \bar{x}.\sigma_i = \bar{x}.\sigma_k\}} \leq T.\sigma_i$ et donc $\hat{\bar{x}}.\sigma_i \leq T.\sigma_i$. Ceci est valable pour tout i donc on a $\underline{\hat{\bar{x}}.\sigma_{c(i)} \leq T.\sigma_{c(i)}}$. De ces deux inégalités, on en déduit que $\underline{\hat{\bar{x}}.\sigma_{c(i)} \leq T.\sigma_i}$.

On reprend la définition de $d_{\bar{x}}$. Si la variable stabilité est égal à $+\infty$ alors la propriété est vérifiée. Sinon on a $\forall i \in \mathbb{N}$:

$$\begin{array}{ll} & d_{\bar{x}}.\sigma_i = T.\sigma_{min\{j|j \geq i \ \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} - \hat{\bar{x}}.\sigma_i \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & d_{\bar{x}}.\sigma_i + \hat{\bar{x}}.\sigma_i = T.\sigma_{min\{j|j \geq i \ \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} \end{array}$$

On a par définition :

$$i < min\{j|j \geq i \ \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}$$

Donc comme T est croissant, en tout i on a :

$$\begin{array}{ll} & T.\sigma_i \leq T.\sigma_{min\{j|j \geq i \ \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} \\ \Rightarrow & \quad \{\text{substitution}\} \\ & T.\sigma_i \leq \hat{\bar{x}}.\sigma_i + d_{\bar{x}}.\sigma_i \end{array}$$

L'égalité n'est pas possible sinon on aurait deux occurrences $\bar{x}$ différentes en deux états ayant la même date. Or ce n'est pas possible pour une exécution séparable. On a donc :

$$\underline{T.\sigma_i < \hat{\bar{x}}.\sigma_i + d_{\bar{x}}.\sigma_i}$$

Ceci est valable pour $c(i)$ donc on a :

$$\underline{T.\sigma_{c(i)} < \hat{\bar{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)}}$$

D'après la définition de $\hat{\bar{y}}$ on a pour tout i :

$$\hat{\bar{y}}.\sigma_i \triangleq T.\sigma_{min\{j|\forall k \in [j..i]:\ \bar{y}.\sigma_i = \bar{y}.\sigma_k\}}$$

Donc il existe un état j tel que $T.\sigma_j = \hat{\bar{y}}.\sigma_i$ et avec $\bar{y}.\sigma_i = \bar{y}.\sigma_j$. Étant donné que dans les deux états, on est dans la même occurrence de y, c'est la même occurrence de x qui a été utilisée pour les deux états i et j d'après la propriété 21. On a donc : $\hat{\bar{x}}.\sigma_{c(j)} = \hat{\bar{x}}.\sigma_{c(i)}$. Or $\hat{\bar{x}}.\sigma_{c(j)} \leq T.\sigma_j$ donc on a $\underline{\hat{\bar{x}}.\sigma_{c(i)} \leq \hat{\bar{y}}.\sigma_i}$. □

La proposition 27 permet notamment d'affirmer que tout chemin P vérifie la propriété suivante :

$$P \left\{ \begin{array}{l} Latency(0,+\infty), Lag(0,+\infty), Stability(0,+\infty), \\ Freshness(0,+\infty), Fitness(0,+\infty), Shift(0,+\infty) \end{array} \right\}$$

En effet, les inégalités de la proposition 27 correspondent aux différences utilisées pour définir chacun des prédicats paramétrant un chemin d'observation. Par exemple, de l'inégalité $\hat{\hat{x}}.\sigma_{c(i)} \leq T.\sigma_i$, on déduit que $0 \leq T.\sigma_i - \hat{\hat{x}}.\sigma_{c(i)} < +\infty$ prouvant que le prédicat de latence avec les bornes 0 et $+\infty$ est bien vérifiée. On rajoute à cela que pour tout état σ_i, on a $d_{\bar{x}}.\sigma_i \in [0..+\infty]$ par définition.

Dans le cadre d'un chemin de propagation sur lequel des propriétés sont définies, on donne une proposition permettant de déduire une forme normale équivalente. Il s'agit de garder dans la définition des propriétés d'un chemin un seul prédicat de chaque type, paramétré par les bornes les plus strictes.

Proposition 28. *Étant donné un chemin* P *entre deux variables* y *et* x*, chemin paramétré par un ensemble de prédicats* $\{\mathtt{Predicate}_1(\delta_1,\Delta_1),\ldots,\mathtt{Predicate}_n(\delta_n,\Delta_n)\}$*, alors on a :*

$$\sigma \vDash P\ \{\mathtt{Predicate}_1(\delta_1,\Delta_1),\ldots,\mathtt{Predicate}_n(\delta_n,\Delta_n)\} \Leftrightarrow$$
$$\sigma \vDash P\ \left\{\begin{array}{l} \mathtt{Latency}(\delta_{Lat},\Delta_{Lat}),\mathtt{Lag}(\delta_{Lag},\Delta_{Lag}),\mathtt{Shift}(\delta_{Shi},\Delta_{Shi}), \\ \mathtt{Freshness}(\delta_{Fre},\Delta_{Fre}),\mathtt{Fitness}(\delta_{Fit},\Delta_{Fit}),\mathtt{Stability}(\delta_{Sta},\Delta_{Sta}) \end{array}\right\}$$

où :

$$\begin{aligned}
\delta_{Lat} &= \max(\{0\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Latency}\}) \\
\Delta_{Lat} &= \min(\{+\infty\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Latency}\}) \\
\delta_{Lag} &= \max(\{0\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Lag}\}) \\
\Delta_{Lag} &= \min(\{+\infty\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Lag}\}) \\
\delta_{Shi} &= \max(\{0\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Shift}\}) \\
\Delta_{Shi} &= \min(\{+\infty\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Shift}\}) \\
\delta_{Fre} &= \max(\{0\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Freshness}\}) \\
\Delta_{Fre} &= \min(\{+\infty\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Freshness}\}) \\
\delta_{Fit} &= \max(\{0\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Fitness}\}) \\
\Delta_{Fit} &= \min(\{+\infty\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Fitness}\}) \\
\delta_{Sta} &= \max(\{0\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Stabiliy}\}) \\
\Delta_{Sta} &= \min(\{+\infty\} \cup \{\delta_i | \mathtt{Predicate}_i = \mathtt{Stabilty}\})
\end{aligned}$$

Démonstration. Pour les prédicats déjà utilisés pour définir les propriétés du chemin de propagation P, on ne garde que les bornes les plus strictes. Dans le cas où il n'y a pas de bornes, on remplace les bornes inférieures par 0 et les bornes supérieures par $+\infty$. D'après la proposition 27, rajouter ces bornes ne modifie pas les propriétés du chemin P puisque ces bornes sont respectées pour tout chemin de propagation bien défini. □

Par la suite, on utilise des propriétés sous leur forme normale.

On donne finalement des propriétés entre les différents prédicats utilisés pour définir un chemin.

Proposition 29. *Soit un ensemble d'observations d'occurrences* $\bar{O}bs$ *et un chemin* P *d'une variable* x *vers une variable* y*, bien défini par rapport à* $\bar{O}bs$*. Alors pour une exécution* σ*, on a :*

$$\begin{array}{lcl}
\sigma \vDash P\ \{\mathtt{Latency}(0,\Delta)\} & \Rightarrow & \sigma \vDash P\ \{\mathtt{Lag}(0,\Delta)\} \\
\sigma \vDash P\ \{\mathtt{Latency}(0,\Delta)\} & \Rightarrow & \sigma \vDash P\ \{\mathtt{Freshness}(0,\Delta)\} \\
\sigma \vDash P\ \{\mathtt{Latency}(0,\Delta)\} & \Rightarrow & \sigma \vDash P\ \{\mathtt{Shift}(0,\Delta)\} \\
\sigma \vDash P\ \{\mathtt{Lag}(\delta,+\infty)\} & \Rightarrow & \sigma \vDash P\ \{\mathtt{Latency}(\delta,+\infty)\} \\
\sigma \vDash P\ \{\mathtt{Freshness}(\delta,+\infty)\} & \Rightarrow & \sigma \vDash P\ \{\mathtt{Latency}((\delta,+\infty)\} \\
\sigma \vDash P\ \{\mathtt{Shift}(\delta,+\infty)\} & \Rightarrow & \sigma \vDash P\ \{\mathtt{Latency}(\delta,+\infty)\}
\end{array}$$

avec $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N}$*.*

Cet ensemble de propriétés ne prétend pas être exhaustif.

Démonstration. D'après la proposition 27 on a :

$$\begin{array}{ll} & \forall c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : \hat{x}.\sigma_{c(i)} \leq T.\sigma_{c(i)} \leq T.\sigma_i \\ \Rightarrow & \quad \{\text{logique}\} \\ & \forall c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : 0 \leq T.\sigma_i - T.\sigma_{c(i)} \leq T.\sigma_i - \hat{x}.\sigma_{c(i)} \end{array}$$

On en déduit que pour une exécution σ, si l'on a :

$$\begin{array}{ll} & \sigma \vDash P\ \{Latency(0,\Delta)\} \\ \Rightarrow & \quad \{\text{par définition}\} \\ & \exists c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : 0 \leq T.\sigma_i - \hat{x}.\sigma_{c(i)} < \Delta \end{array}$$

Alors on a :

$$\begin{array}{ll} & \exists c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : 0 \leq T.\sigma_i - T.\sigma_{c(i)} < \Delta \\ \Rightarrow & \quad \{\text{par définition}\} \\ & \sigma \vDash P\ \{Shift(0,\Delta\} \end{array}$$

De même, si l'on a :

$$\begin{array}{ll} & \sigma \vDash P\ \{Shift(\delta,+\infty)\} \\ \Rightarrow & \quad \{\text{par définition}\} \\ & \exists c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : \delta \leq T.\sigma_i - T.\sigma_{c(i)} \end{array}$$

Alors on a :

$$\begin{array}{ll} & \exists c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : \delta \leq T.\sigma_i - \hat{x}.\sigma_{c(i)} \\ \Rightarrow & \quad \{\text{par définition}\} \\ & \sigma \vDash P\ \{Latency(\delta,+\infty)\} \end{array}$$

En utilisant la proposition 27, on a de même :

$$\forall c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : 0 \leq T.\sigma_c(i) - \hat{x}.\sigma_{c(i)} \leq T.\sigma_i - \hat{x}.\sigma_{c(i)} \wedge 0 \leq \hat{y}.\sigma_i - \hat{x}.\sigma_{c(i)} \leq T.\sigma_i - \hat{x}.\sigma_{c(i)}$$

De ces inégalités, on déduit de la même manière les propositions restantes. □

Proposition 30. *Soit un ensemble d'observations d'occurrences $\bar{\mathcal{O}}bs$. Alors pour une exécution σ et pour tout entier Δ, on a :*

$$\begin{array}{l} \forall P \in \mathcal{P}aths(\bar{\mathcal{O}}bs) : \forall P_2 \in \mathcal{P}aths(\bar{\mathcal{O}}bs) : P = P_1;P_2;P_3 \Rightarrow \\ (\sigma \vDash P\ \{Latency(0,\Delta)\} \Rightarrow \sigma \vDash P_2\ \{Latency(0,\Delta)\}) \end{array}$$

Pour un chemin P ayant une borne supérieure sur la latence, les sous-chemins de P sont soumis à cette même borne.

Démonstration. Soit un ensemble de variables $x_1, x_2, \ldots x_n$ tel que $[x_1, x_2, \ldots, x_n]$ est bien défini par un ensemble d'observations $\bar{\mathcal{O}}bs$. Soit k, l deux entiers appartenant à $[1..n]$ et tel que $k+1 < l$. Alors $P_1 = [x_1, \ldots, x_{k+1}]$, $P_2 = [x_{k+1}, \ldots, x_l]$ et $P_3 = [x_l, \ldots, x_n]$ sont trois sous-chemins de $[x_1, x_2, \ldots, x_n]$.

Soit $i \in \mathbb{N}$. Soit c' une horloge définie par :

$$\exists c_2, \ldots, c_{k+1} : \forall m \in [2..k+1], \exists f_m, X_{m-1} : (x_m \lessdot f_m(X_{m-1})) \in \bar{\mathcal{O}}bs \wedge x_{m-1} \in X_{m-1} \wedge c_m \in \mathcal{C}(x_m \lessdot f_m(X_{m-1})).\sigma \wedge c' = c_2 \circ \ldots \circ c_{k+1}$$

De même, on définit c'' :

$$\exists c_{k+2}, \ldots, c_l : \forall m \in [k+2..l], \exists f_m, X_{m-1} : (x_m \trianglelefteq f_m(X_{m-1})) \in \bar{\mathcal{O}}bs \wedge x_{m-1} \in X_{m-1} \wedge c_m \in \mathcal{C}(x_m \trianglelefteq f_m(X_{m-1})).\sigma \wedge c'' = c_{k+2} \circ \ldots \circ c_l$$

Et finalement on définit c''' par :

$$\exists c_{l+1}, \ldots, c_n : \forall m \in [l+1..n], \exists f_m, X_{m-1} : (x_m \trianglelefteq f_m(X_{m-1})) \in \bar{\mathcal{O}}bs \wedge x_{m-1} \in X_{m-1} \wedge c_m \in \mathcal{C}(x_m \trianglelefteq f_m(X_{m-1})).\sigma \wedge c''' = c_{l+1} \circ \ldots \circ c_n$$

Ces trois horloges sont définies uniquement comme la composition d'horloges d'observation. D'après la définition 28 des horloges d'un chemin, elles font partie des horloges des chemins P_1, P_2 et P_3. On suppose que pour la jonction de ces trois chemins, on a :

$$\begin{array}{ll} & \sigma \vDash P_1; P_2; P_3 \ \{Latency(0, \Delta)\} \\ \Leftrightarrow & \quad \{\text{par définition}\} \\ & \forall i \in \mathbb{N}, \exists c \in \mathcal{C}(P_1; P_2; P_3).\sigma : T.\sigma_i - \hat{\mathbb{x}}_1.\sigma_{c(i)} < \Delta \end{array}$$

On utilise un raisonnement par l'absurde, on suppose donc que l'on n'a pas $\sigma \vDash P_2 \ \{Latency(0, \Delta)\}$. Dans ce cas, on a :

$$\begin{array}{ll} & \exists j \in \mathbb{N}, \forall c \in \mathcal{C}(P_2).\sigma : T.\sigma_j - \hat{\mathbb{x}}_{k+1}.\sigma_{c(j)} \geq \Delta \\ \Rightarrow & \quad \{c'' \in \mathcal{C}(P_2).\sigma\} \\ & \exists j \in \mathbb{N} : T.\sigma_j - \hat{\mathbb{x}}_{k+1}.\sigma_{c''(j)} \geq \Delta \end{array}$$

On a $c'''(j) \leq j$ et donc $c''(c'''(j)) \leq c''(j)$. Donc d'après les propriétés du profil temporel, on a $\hat{\mathbb{x}}_{k+1}.\sigma_{c''(c'''(j))} \leq \hat{\mathbb{x}}_{k+1}.\sigma_{c''(j)}$ et on en déduit que l'on a :

$$\exists j \in \mathbb{N} : T.\sigma_j - \hat{\mathbb{x}}_{k+1}.\sigma_{c''(c'''(j))} \geq \Delta$$

D'après la propriétés 27 appliquées au chemin P_1 avec l'horloge c', on a $\hat{\mathbb{x}}_1.\sigma_{c'(c''(c'''(j)))} \leq \hat{\mathbb{x}}_{k+1}.\sigma_{c''(c'''(j))}$. On a donc :

$$\exists j \in \mathbb{N} : T.\sigma_j - \hat{\mathbb{x}}_1.\sigma_{c'(c''(c'''(j)))} \geq \Delta$$

D'après la proposition 20, pour tout horloge du chemin $P_1; P_2; P_3$ la valeur de $\hat{\mathbb{x}}_1.\sigma_{c(i)}$ est la même. On en déduit donc que :

$$\exists j \in \mathbb{N} : \forall c \in \mathcal{C}(P_1; P_2; P_3).\sigma : T.\sigma_j - \hat{\mathbb{x}}_1.\sigma_{c(j)} \geq \Delta$$

Ce qui contredit la propriété de latence de $P_1; P_2; P_3$. On en déduit donc la latence de P_2.

$$T.\sigma_i - \hat{\mathbb{x}}_2.\sigma_{c'(i)} < \Delta$$

□

En utilisant un raisonnement par l'absurde, on a prouvé au passage la proposition suivante :

Proposition 31. *Soit un ensemble d'observations d'occurrences $\bar{\mathcal{O}}bs$. Alors pour une exécution σ et pour tout entier δ, on a :*

$$\forall P \in \mathcal{P}aths(\bar{\mathcal{O}}bs) : \exists P_2 \in \mathcal{P}aths(\bar{\mathcal{O}}bs) : P = P_1; P_2; P_3 \Rightarrow \sigma \vDash P_2 \ \{Latency(\delta, +\infty)\} \Rightarrow \sigma \vDash P \ \{Latency(\delta, +\infty)\}$$

Si un chemin P a un de ces sous-chemins avec une borne minimum sur la latence, alors P a aussi cette borne sur sa latence.

4.2.5 *Influence des propriétés des différents chemins*

Pour un chemin de propagation, on peut donner une propriété sur la propagation des occurrences le long de ce chemin mais aussi des propriétés sur des sous-chemins ou des sur-chemins. Ces différentes propriétés ne sont pas totalement indépendantes. On a par exemple vu dans les propositions 30 et 31 que la propriété de latence d'un chemin implique des propriétés de latence des sur-chemins et réciproquement. D'autres propriétés similaires pourraient être données.

Définir les propriétés de différents chemin à partir des mêmes horloges pour vérifier toutes les propriétés eut été possible. Pour un chemin, on peut imaginer devoir choisir une horloge qui est la composition des horloges vérifiant les propriétés des différents sous-chemins. Outre la difficulté pour donner une telle définition de manière concise, cela semble être un lien trop fort entre les différentes propriétés. Nous préférons pouvoir exprimer chacune des propriétés indépendamment. C'est pour cela que les propriétés de deux chemins sont définis par des horloges indépendantes.

4.3 SPÉCIFICATION D'UN SYSTÈME

4.3.1 *Définition d'une spécification*

On définit la spécification d'un système comme un couple. Le premier élément est l'architecture du système et décrit donc la propagation des valeurs des différentes variables dans le système et les liens entre les variables. Le deuxième élément est l'ensemble des propriétés temps réel du système qui limitent le comportement des variables et la propagation de leurs valeurs.

Définition 39. *Spécification d'un système. La spécification d'un système est définie par un couple* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop} \rangle$ *tel que* $\mathcal{A}\mathtt{rchi}$ *est un ensemble de relations d'observation d'occurrences et* $\mathcal{P}\mathtt{rop}$ *un ensemble de propriétés sous forme normale sur les variables ou les chemins de propagation de l'architecture* $\mathcal{A}\mathtt{rchi}$*. L'ensemble des exécutions définies par la sémantique opérationnelle des relations d'observations et des propriétés temporelles associées est défini par :*

$$[\![\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop} \rangle]\!]_{\mathcal{O}} \triangleq \{\sigma | \sigma \in [\![\mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop}]\!]_{\mathcal{O}} \ \wedge \ \sigma \vDash \mathcal{P}\mathtt{rop}\}$$

Définition 40. *Une spécification* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop} \rangle$ *est bien formée si :*
- *chaque variable de l'architecture est au plus image d'une observation ;*
- *on a au plus une seule propriété par variable de l'architecture et une seule propriété par chemin de propagation ;*
- *les propriétés sur les chemins sont définies sur les chemins de l'architecture* $\mathcal{P}\mathtt{aths}(\mathcal{A}\mathtt{rchi})$.

Pour une telle spécification, chacune des variables ne peut être image que d'une seule observation. La définition originale de l'observation n'empêche pas une variable d'être image de deux observations. Dans ce cas, les sources doivent avoir un historique de valeur commun et c'est cet historique commun qui doit être extrait par l'horloge d'observation. On a donc deux observations fortement synchronisées.

Une spécification est donc bien définie si pour chaque élément de l'architecture il y a au plus une propriété constituée d'une conjonction de prédicats temporels.

Il faut finalement que les propriétés des chemins soient définies sur des chemins de l'ensemble $\mathcal{P}\mathtt{aths}(\mathcal{A}\mathtt{rchi})$. Il s'agit des chemins bien définis par l'architecture et ne comportant pas de boucle. Les chemins dans le graphe de l'architecture repassant par

un même noeud ne peuvent donc pas être utilisés pour définir de propriétés sur les chemins.

On s'intéresse au cas des boucles de causalité de l'architecture. Il s'agit des chemins définis par le graphe de l'architecture de longueur strictement supérieure à 1 allant d'une variable x à cette même variable. On veut qu'il n'y ait pas le long de la boucle un décalage logique nul. Un décalage nul impose en effet que toutes les variables soient mises à jour simultanément de manière synchrone. On ne s'intéresse pas aux spécifications ayant une telle boucle.

Définition 41. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$. Cette spécification est libre de boucle de causalité synchrone si jamais pour tout chemin P de longueur strictement supérieure à 1 d'une variable x à cette même variable x on a :*

$$\exists P_1, P_2, P_3 \in \mathcal{P}aths(\mathcal{A}rchi), \exists \delta \in \mathbb{N}, \exists \Delta \in \mathbb{N} \cup \{+\infty\} :$$
$$P = [x]; P_1; P_2; P_3; [x] \ \wedge \ P_2 \ \{Shift(\delta, \Delta)\} \in \mathcal{P}rop \ \wedge \ \delta > 0$$

Dans chaque boucle, il y a donc au moins un maillon qui entraîne un décalage supérieur à 1. Ceci entraîne alors d'après les propositions 29 et 31 d'avoir une latence minimum de 1 le long de ce chemin.

Par la suite, on ne considère que des spécifications sous forme normale et sans boucle de causalité synchrone. On étudie notamment la satisfiabilité d'une telle spécification que l'on définit ici :

Définition 42. *Satisfaction d'une spécification. Soit une spécification d'un système $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$. On note $\sigma \vDash \langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ la satisfaction de la spécification par une exécution. Cette satisfaction est définie par :*

$$\sigma \vDash \langle \mathcal{A}rchi, \mathcal{P}rop \rangle \triangleq \forall obs \in \mathcal{A}rchi : \sigma \vDash obs \ \wedge \ \forall prop \ \in \mathcal{P}rop : \sigma \vDash prop$$

La spécification est donc satisfaite par une exécution si chacune des observations de l'architecture est satisfaite et si chacune des propriétés l'est aussi.

4.3.2 *Caractéristiques d'une spécification*

Définition 43. *Paramètres des propriétés de propagation sur les chemins. Pour une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, on définit la fonction $\mathtt{PathParameters}$ pour un chemin telle que :*

$$\forall P \in \mathcal{P}aths(\mathcal{A}rchi) : \mathtt{PathParameters}(P) \triangleq$$
$$\left\{ \begin{array}{l} [\delta_{Lat}, \Delta_{Lat}, \delta_{Lag}, \Delta_{Lag}, \delta_{Shi}, \Delta_{Shi}, \delta_{Fre}, \Delta_{Fre}, \delta_{Fit}, \Delta_{Fit}, \delta_{Sta}, \Delta_{Sta}] \ \textit{si} \\ P \left\{ \begin{array}{l} Latency(\delta_{Lat1}, \Delta_{Lat1}), Lag(\delta_{Lag1}, \Delta_{Lag1}), Shift(\delta_{Shi1}, \Delta_{Shi1}), \\ Freshness(\delta_{Fre1}, \Delta_{Fre1}), Fitness(\delta_{Fit1}, \Delta_{Fit1}), Stability(\delta_{Sta1}, \Delta_{Sta1}) \end{array} \right\} \in \mathcal{P}rop \\ [0, +\infty, 0, +\infty, 0, +\infty, 0, +\infty, 0, +\infty, 0, +\infty] \ \textit{sinon} \end{array} \right.$$

Cet ensemble d'entiers caractérise les propriétés données par la spécification sur un chemin. On s'intéresse aussi aux paramètres sur chacun des prédicats.

Définition 44. *Paramètres des propriétés de propagation sur les chemins. Pour une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, on définit la fonction $\mathtt{MinLatency}$ pour un chemin telle que :*

$$\forall P \in \mathcal{P}aths(\mathcal{A}rchi) : \mathtt{MinLatency}(P) \triangleq$$
$$\left\{ \begin{array}{ll} \delta_i & \textit{si } \exists i \in [1..n] : \exists P \ \{Predicate_1(\delta_1, \Delta_1), \ldots, Predicate_n(\delta_n, \Delta_n)\} \in \mathcal{P}rop : \\ & Predicate_i = Latency \\ 0 & \textit{sinon} \end{array} \right.$$

On définit la fonction $\mathtt{MaxLatency}$ *par :*

$$\forall P \in \mathcal{P}aths(\mathcal{A}rchi) : \mathtt{MaxLatency}(P) \triangleq$$
$$\begin{cases} \Delta_i & \textit{si } \exists i \in [1..n] : \exists P\ \{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\} \in \mathcal{P}rop : \\ & \mathtt{Predicate}_i = \mathtt{Latency} \\ +\infty & \textit{sinon} \end{cases}$$

De la même manière, on définit les fonctions $\mathtt{MinLag}$, $\mathtt{MaxLag}$, $\mathtt{MinShift}$, ...

Dans la définition donnée, il s'agit de la borne minimum du prédicat de latence paramétrant le chemin P. Cette valeur est unique pour une spécification bien formée.

Ces définitions ont pour but d'alléger les définitions et les propriétés données dans la suite.

5

RÉGIME INITIAL

Nous allons voir que certaines propriétés sont impossibles à satisfaire au démarrage du système, c'est-à-dire dans l'état initial et sur un préfixe fini. Pour autant nous allons montrer qu'il existe une borne calculable pour certaines spécifications au delà de laquelle la spécification peut et doit être vérifiée.

On commence par montrer pourquoi certaines propriétés sont impossibles à satisfaire au démarrage du système. On redéfinit ensuite les propriétés d'une spécification afin de prendre en compte ce problème. Pour cela, on définit un régime initial pour les variables et les chemins d'une spécification, régime initial pendant lequel les définitions des propriétés d'une spécification sont modifiées afin de pouvoir être vérifiées. On choisit alors, en fonction des propriétés sur les variables et les chemins, des conditions sur la sortie de ce régime initial. On montre finalement que pour certaines spécifications, ces conditions impliquent une sortie du régime initial en temps borné et calculable.

5.1 INTRODUCTION DU PROBLÈME ET DE SA SOLUTION

Le temps, dans un système défini par un ensemble de relations d'observation et de propriétés temporelles sur ces relations, est initialisé à 0. Il en est de même pour les différentes horloges des observations et celles caractérisant le décalage le long d'un chemin. On considère une spécification définissant un chemin P entre deux variables x et y et la propriété suivante :

$$P\ \{\mathtt{Shift}(2,5)\}$$

Pour une exécution σ vérifiant cette propriété, il doit exister une horloge c du chemin P tel qu'en tout état i :

$$2 \leq T.\sigma_i - T.\sigma_{c(i)} < 5$$

En l'état initial, on a donc :

$$2 \leq 0 - 0 < 5$$

Le prédicat de décalage $\mathtt{Shift}$ n'étant pas vérifié à l'instant initial, cette exécution ne vérifie pas la spécification et aucune autre ne la vérifie. Cette spécification n'est donc pas satisfiable. De même, d'autres propriétés temporelles sur un chemin ne peuvent pas être satisfaites dans l'état initial et sur un préfixe fini. On sait qu'à l'instant initial on a pour toute variable x : $\hat{x}.\sigma_0 = 0$. Pour un chemin P entre deux variables x et y, les inégalités suivantes, pour une valeur de δ strictement positive, ne sont pas satisfiables :

$$\delta \leq T.\sigma_i - \hat{x}.\sigma_{c(i)}$$
$$\delta \leq \hat{y}.\sigma_i - \hat{x}.\sigma_{c(i)}$$
$$\delta \leq T.\sigma_i - T.\sigma_{c(i)}$$
$$\delta \leq T.\sigma_{c(i)} - \hat{x}.\sigma_{c(i)}$$

Les propriétés de latence, de lag, de fraîcheur ou de décalage ne sont donc pas respectées en l'état initial lorsqu'un minimum autre que 0 les caractérise. Il existe un préfixe fini à

toute exécution dans lequel ces propriétés ne sont pas vérifiables. Toute spécification comportant de telles propriétés est donc non satisfiable.

Les propriétés temporelles sur les chemins ne sont pas les seules à poser ce problème. On considère une variable x et une propriété de sporadicité telle que :

$$x\ \{\mathtt{Sporadic}(2,5)\}$$

Par définition, dans l'état initial on a $\hat{\hat{x}}.\sigma_0 = 0$. D'après la sporadicité, on ne pourra pas avoir de mise à jour avant que deux unités de temps se soient écoulées. Dans un système réel, il peut y avoir une phase telle que la première mise à jour n'a pas nécessairement lieu dès le démarrage du système. En forçant la première mise à jour à être à l'instant 0, on se réduit à une seul phase initiale possible. On considère donc uniquement un sous-ensemble des exécutions du système que l'on veut modéliser.

De même, la mise à jour de l'image d'une observation dépend de la disponibilité de nouvelles occurrences des sources. Pour pouvoir effectuer la première mise à jour d'une telle variable, des occurrences des sources doivent être disponibles. Il ne peut donc pas y avoir de mise à jour de l'image tant que les sources n'ont pas été mises à jour. Dans un système réactif classique, avec des calculs périodiques, le temps que les valeurs des entrées se propagent, les premiers calculs effectués par certains composants sont basés sur les valeurs d'initialisation des entrées. Ces calculs, que l'on peut qualifier de non pertinents, ne définissent pas de nouvelles occurrences des images d'après la proposition 15 et ne donnent donc pas de mises à jour des images. Ce n'est que lorsque des occurrences des sources, remplaçant les occurrences initiales, seront propagées dans le système que les résultats des calculs seront pertinents. C'est ce problème qui est traduit par l'attente de la disponibilité d'occurrence des sources pour pouvoir mettre à jour l'image d'une observation.

Afin de prendre en compte ces problèmes dans les premiers états d'une exécution, on définit un régime initial pour chaque variable et pour les chemins d'une architecture.

Définition 45. *Régime initial. Soit une spécification $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop}\rangle$ et une exécution σ vérifiant cette spécification. On définit le régime initial de différents éléments de cette spécification.*

- *Une variable x de l'architecture est dans le régime initial tant qu'elle n'a pas été mise à jour et donc pour tout état σ_i tel que $\hat{\hat{x}}.\sigma_i$ est égal à 0.*
- *Un chemin P d'une variable x vers une variable y est dans le régime initial en tout état σ_i tant que pour tout horloge c de ce chemin, on a : $\hat{\hat{x}}.\sigma_{c(i)} = 0$.*
- *Une spécification est dans le régime initial tant qu'il existe une variable ou un chemin de l'architecture dans le régime initial.*

Le régime initial est un préfixe de l'exécution pendant lequel la définition des propriétés de la spécification est modifiée pour pouvoir être vérifiées. Il s'agit pour les variables de prendre en compte la phase initiale avant la première mise à jour et la disponibilité d'occurrence des sources. Pour les chemins, il s'agit de prendre en compte le temps de propagation des premières occurrences des sources le long de ce chemin.

La sortie du régime initial d'un chemin entraîne la sortie du régime initial de l'image de ce chemin et nécessite la sortie du régime initial de la source du chemin. Il suffit donc que tous les chemins sortent du régime initial pour que la spécification sorte de son régime initial. L'inverse est faux, une variable image de plusieurs chemins peut être mise à jour avec une nouvelle occurrence d'une source d'un seul de ces chemins. Seul ce chemin sort alors de son régime initial.

On veut borner le temps pendant lequel certaines variables et certains chemins sont dans le régime initial. Les bornes sont choisies selon les propriétés de la spécification.

On montre alors que ces choix permettent de calculer une borne sur le temps pendant lequel toutes les variables et les chemins du systèmes restent dans le régime initial.

5.2 REDÉFINITION DES PROPRIÉTÉS D'UNE SPÉCIFICATION

5.2.1 *Régime initial d'une variable*

Une variable est dans le régime initial tant qu'elle n'a pas été mise à jour. On veut avoir un temps maximum avant cette première mise à jour.

Objectif 1. *Temps maximum avant la première mise à jour. Soit une architecture $\mathcal{A}\mathrm{rchi}$ et un ensemble de propriétés temps réel $\mathcal{P}\mathrm{rop}$. On veut qu'il existe un temps maximum avant la première mise à jour d'une variable x défini comme un entier Δ_x tel que pour toute exécution σ vérifiant la spécification il y ait :*

$$\forall i \in \mathbb{N} : T.\sigma_i \geq \Delta_x \Rightarrow \hat{\hat{x}}.\sigma_i \neq 0$$

Un tel temps maximum dépendrait, dans le cas de l'image d'une observation, de la disponibilité d'occurrences des sources d'une observation. La proposition 15 page 57 nous dit que pour que $\hat{\hat{y}}$ évolue, il faut qu'une nouvelle occurrence d'une source soit utilisée pour définir la nouvelle occurrence de y. Dans le cas d'une variable non image d'une observation, un tel temps dépend de la phase initiale du système.

On redéfinit les propriétés sur le comportement d'une variable afin d'introduire le régime initial et le temps maximum avant la première mise à jour.

SPORADICITÉ D'UNE VARIABLE. On redéfinit la sporadicité d'une variable en introduisant le temps maximum avant la première mise à jour de cette variable. Cette première mise à jour détermine la phase initiale.

Définition 46. *Sporadicité. Soit une exécution séparée σ et une variable x dont le temps maximum avant la première mise à jour est Δ_x. Le comportement de x lors de cette exécution est sporadique de paramètres δ, Δ si on a :*

$$\sigma \vDash x\ \{\mathrm{Sporadic}(\delta, \Delta)\} \triangleq \forall i \in \mathbb{N} : (d_{\hat{x}}.\sigma_i \in [\delta..\Delta] \ \vee\ (T.\sigma_i < \Delta_x \ \wedge\ \hat{\hat{x}}.\sigma_i = 0))$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$.

Tant que l'on n'a pas atteint Δ_x ou qu'il n'y a pas eu de première mise à jour, la propriété de sporadicité telle qu'elle était définie à l'origine n'a pas à être respectée. A partir du moment où une mise à jour s'est produite, la durée entre deux mises à jour doit respecter la propriété de la sporadicité, quel que soit l'instant de cette mise à jour. Par définition, cette première mise à jour survient nécessairement avant Δ_x.

PÉRIODICITÉ D'UNE VARIABLE On redéfinit la périodicité pour introduire le régime initial et un temps maximum avant la première mise à jour d'une variable. Comme pour la sporadicité, avant la première mise à jour la propriété de périodicité d'origine n'est pas vérifiée. Cette première mise à jour vient avant ce temps maximum et on vérifie ensuite la propriété de périodicité d'origine. Pour respecter le sens de la propriété de périodicité, la première mise à jour doit tout de même respecter la période et donc survenir dans un intervalle autorisé par la périodicité. On complète de plus la définition de la propriété de périodicité avec une phase. Il s'agit de décaler les instants autour desquels les mises à jour ont lieu. On a donc la définition suivante :

Définition 47. *Périodicité. Soit une exécution séparée* σ *et une variable* x *dont le temps maximum avant la première mise à jour est* Δ_x*. Le comportement de* x *lors de cette exécution est périodique de période* P*, de gigue* J *et de phase* ϕ *si on a :*

$$\sigma \vDash x\ \{\mathtt{Periodic}(P, J, \phi)\} \triangleq \exists k \in \mathbb{N}, \exists \delta = \langle \delta_i | i \geq k \wedge \delta_i \in [-J..J] \rangle :$$
$$\{\hat{x}.\sigma_i | i \in \mathbb{N}\} = \{n * P + \phi + \delta_n | n \in \mathbb{N} \setminus [0..k-1]\} \cup \{0\} \wedge\ (k * P + \phi + \delta_k \leq \Delta_x)$$

avec $J < P/2$*,* $\phi < P$ *et si* $k = 0$ *alors* $\delta_0 \in [\min(-\phi, -J)..J]$*.*

Dans cette définition, le choix de k détermine l'instant de la première mise à jour c'est-à-dire le premier intervalle autour des $(n * P + \phi)$ où il y a une mise à jour. Après cette mise à jour il y en a une à chaque période. La phase fait que les instants autour desquels ont lieu les mises à jour sont de la forme $k * P + \phi$ et non pas $k * P$.

Malgré la nouvelle définition des propriétés de périodicité et de sporadicité, la proposition 23 donnant le lien entre périodicité et sporadicité est toujours vraie. En effet, dans les deux cas la première mise à jour doit survenir avant que le temps maximum soit atteint. Une fois la première mise à jour effectuée, l'écart entre deux mises à jour périodiques est la même qu'avec la définition originale et ce malgré l'introduction de la phase. La propriété de sporadicité est donc respectée par une variable périodique.

5.2.2 *Régime initial d'un chemin de propagation*

Comme pour une variable, on veut avoir un temps maximum de sortie du régime initial d'un chemin.

Objectif 2. *Temps maximum de sortie du régime initial d'un chemin. Soit une architecture* $\mathcal{A}\mathrm{rchi}$ *et un ensemble de propriétés temps réel* $\mathcal{P}\mathrm{rop}$*. Pour tout chemin* P *d'une variable* x *vers une variable* y *bien défini par* $\mathcal{A}\mathrm{rchi}$ *on veut qu'il existe un temps maximum de sortie du régime initial de ce chemin* P *défini comme un entier* Δ_P *et tel que pour tout exécution* σ *vérifiant la spécification on a :*

$$\forall i \in \mathbb{N} : \forall c \in \mathcal{C}(P).\sigma : T.\sigma_i \geq \Delta_P \Rightarrow \hat{x}.\sigma_{c(i)} \neq 0$$

L'objectif est qu'il existe un temps maximum avant que le chemin sorte du régime initial. Le chemin sort de son régime initial lorsque l'horloge de ce chemin pointe vers une occurrence autre que l'occurrence initiale de la source. On veut donc que le chemin sorte du régime initial dès qu'il y a eu une mise à jour de la source et que l'occurrence issue de cette mise à jour s'est propagée le long du chemin et est utilisée pour mettre à jour l'image.

PROPRIÉTÉS SUR LA PROPAGATION DES OCCURRENCES Les propriétés d'un chemin, telles que définies précédemment, peuvent ne pas être satisfiables dans les premiers instants d'une exécution. On modifie leur définition pour tenir compte du régime initial et introduire le temps maximum de sortie du régime initial. Les propriétés n'ont alors pas à être vérifiées lors des premiers instants d'une exécution et tant que le chemin n'est pas sorti du régime initial.

Définition 48. *Lag. Pour une exécution* σ*, le prédicat de lag pour un état* σ_i *et pour un chemin* P *d'une variable* x *vers une variable* y *est défini par :*

$$\mathtt{Lag}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq \hat{y}.\sigma_i - \hat{x}.\sigma_{c(i)} < \Delta \vee (T.\sigma_i < \Delta_P \wedge \hat{x}.\sigma_{c(i)} = 0)$$

et où $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$ *et* $\delta < \Delta$*.*

Tant qu'il n'y a pas eu de première mise à jour ou tant qu'on n'a pas dépassé le temps de sortie du régime initial, on est dans le régime initial et donc le lag n'a pas à être vérifié. On donne une modification semblable pour les autres prédicats utilisés pour définir les propriétés d'un chemin. On a donc :

Définition 49. *Stabilité. Pour une exécution σ, le prédicat de stabilité pour un état σ_i et pour un chemin P d'une variable x vers une variable y est défini par :*

$$\mathtt{Stability}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq d_{\bar{x}}.\sigma_{c(i)} < \Delta \vee (T.\sigma_i < \Delta_P \wedge \hat{\bar{x}}.\sigma_{c(i)} = 0)$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$ et $\delta < \Delta$.

Définition 50. *Fraîcheur. Pour une exécution σ, le prédicat de fraîcheur pour un état σ_i et pour un chemin P d'une variable x vers une variable y est défini par :*

$$\mathtt{Freshness}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq T.\sigma_{c(i)} - \hat{\bar{x}}.\sigma_{c(i)} < \Delta \vee (T.\sigma_i < \Delta_P \wedge \hat{\bar{x}}.\sigma_{c(i)} = 0)$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$ et $\delta < \Delta$.

Définition 51. *Pérennité. Pour une exécution σ, le prédicat de pérennité pour un état σ_i et pour un chemin P d'une variable x vers une variable y est définie par :*

$$\mathtt{Fitness}(P, c, \delta, \Delta).\sigma_i \triangleq \delta < \hat{\bar{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - T.\sigma_{c(i)} \leq \Delta \vee (T.\sigma_i < \Delta_P \wedge \hat{\bar{x}}.\sigma_{c(i)} = 0)$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$ et $\delta < \Delta$.

Définition 52. *Décalage. Pour une exécution σ, le prédicat de décalage pour un état σ_i et pour un chemin P d'une variable x vers une variable y et tel que le temps maximum avant la première mise à jour de x est Δ_x est défini par :*

$$\mathtt{Shift}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq T.\sigma_i - T.\sigma_{c(i)} < \Delta \vee (T.\sigma_i < \Delta_P \wedge \hat{\bar{x}}.\sigma_{c(i)} = 0)$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$ et $\delta < \Delta$.

Définition 53. *Latence. Pour une exécution σ, le prédicat de latence pour un état σ_i et pour un chemin P d'une variable x vers une variable y et tel que le temps maximum avant la première mise à jour de x est Δ_x est défini par :*

$$\mathtt{Latency}(P, c, \delta, \Delta).\sigma_i \triangleq \delta \leq T.\sigma_i - \hat{\bar{x}}.\sigma_{c(i)} < \Delta \vee (T.\sigma_i < \Delta_P \wedge \hat{\bar{x}}.\sigma_{c(i)} = 0)$$

et où $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$ et $\delta < \Delta$.

Malgré les changements de définitions des différents prédicats, les propriétés 29, 30 et 31 sont toujours vérifiées. En effet, concernant la propriété 29, on vérifie bien que les conditions rajoutées dans chaque prédicat implique celles ajoutées pour les autres prédicats puisque ce sont les mêmes. Concernant les propriétés 30 et 31, il n'est pas possible qu'un chemin sorte du régime initial sans que les sous-chemins en soient sortis, le temps de propagation le long du sous-chemin étant plus court que celui le long du chemin. En utilisant cette remarque on prouve que les deux propriétés restent justes.

5.2.3 *Régime initial d'une spécification*

On s'intéresse à la sortie du régime initial d'une spécification.

Définition 54. *Temps maximum de sortie du régime initial d'une spécification. Soit une spécification* $S = \langle \mathcal{A}rchi, \mathcal{P}rop \rangle$*. On définit le temps maximum de sortie du régime initial de cette spécification par un entier* Δ_S *égal à :*

$$\Delta_S = max(\{\Delta_P | P \in \mathcal{P}aths(\mathcal{A}rchi)\} \cup \{\Delta_x | x \in \mathcal{V}ar(\mathcal{A}rchi)\})$$

La spécification sort du régime initial lorsque tous les chemins et toutes les variables sont sortis de leur régime initial. On définit donc le temps maximum de sortie du régime initial d'une spécification en fonction des temps de sortie des chemins et des variables de l'architecture.

La sortie du régime initial d'un chemin entraîne la sortie du régime initial de l'image du chemin et nécessite la sortie du régime initial de la source. Lorsque tous les chemins sont sortis de leur régime initial, on est assuré que toutes les variables faisant partie d'un chemin sont sorties de leur régime initial.

Lorsque les chemins sont sortis de leur régime initial, ils ont été parcourus par une occurrence d'une variable sortie du régime initial. Alors les occurrences des images des observations de l'architecture sont uniquement définies par des occurrences de variables sources sorties de leur régime initial. Donc toutes les valeurs d'initialisation du système ont été remplacées par des valeurs issues de calcul.

On ne considère pas les chemins contenant une boucle dans cette définition car en créant des chemins avec toujours plus de boucles, on augmente leur temps de sortie du régime initial. On ne pourrait alors pas borner le temps de sortie du régime initial de la spécification.

5.3 CHOIX DU TEMPS DE SORTIE DES RÉGIMES INITIAUX

5.3.1 *Choix du temps de sortie du régime initial d'une variable*

Dans le cas où une variable n'est pas image d'une observation, le comportement de cette variable dépend uniquement des propriétés de sporadicité ou de périodicité. On donne donc une borne sur le temps de sortie du régime initial tirée des propriétés sur le comportement d'une telle variable. On suppose que l'on a une variable x ayant la propriété suivante :

$$x \{Sporadic(\delta, \Delta)\}$$

La propriété de sporadicité indique qu'il ne peut y avoir plus de Δ unités de temps entre deux mises à jour. On considère donc que pour une telle variable, le temps maximum avant la première mise à jour est de Δ.

Choix 1. *Soit une exécution séparée* σ *et une variable* x *non image d'une observation. Si cette variable est sporadique de paramètres* δ, Δ *et donc telle que :*

$$\sigma \vDash x \{Sporadic(\delta, \Delta)\}$$

alors pour une telle variable, le temps maximum avant sa première mise à jour Δ_x *est égal à* Δ *la borne supérieure de la sporadicité.*

Justification. Pour une telle variable, la première mise à jour ne dépend pas de la disponibilité d'occurrence. Seul la sporadicité influence le comportement de la variable. La phase initiale ne devrait pas excéder le temps maximum entre deux mises à jour, d'où le choix effectué ici.

Pour une variable périodique, non image d'une observation, on donne aussi un temps de sortie du régime initial définie par la propriété sur le comportement de cette variable.

Choix 2. *Soit une exécution séparée* σ *et une variable* x *non image d'une observation. On suppose que cette variable est périodique de paramètres* P, J, ϕ *et donc telle que*

$$\sigma \vDash x \ \{\mathtt{Periodic}(P, J, \phi)\}$$

Alors pour une telle variable, le temps maximum avant sa première mise à jour Δ_x *est égal à :*

$$\Delta_x = \phi + J \textit{ si } \phi > J$$

et sinon à :

$$\Delta_x = P + \phi + J \textit{ si } \phi \leq J$$

Démonstration. Comme pour la sporadicité, c'est une interprétation de la périodicité qui nous donne cette proposition. La phase initiale est définie selon la phase définissant la propriété de périodicité.

Dans le cas où l'on $\phi \leq J$ l'intervalle autour de ϕ n'est pas entièrement supérieur à 0 et donc la gigue n'est pas libre. Donc on attend la période suivante. Si par contre $\phi \leq J$ alors, toute les gigues sont possibles. □

On donne désormais le temps maximum de sortie du régime initial pour une variable image d'une observation.

Choix 3. *Temps maximum avant la première mise à jour. Soit une spécification* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop} \rangle$. *Le temps maximum avant la première mise à jour d'une variable* y *image d'une observation* $y \lessdot f(X)$ *est choisi égal à :*

$$\Delta_y = \min\{\Delta_P | P \in \mathcal{P}\mathtt{aths}(\mathcal{A}\mathtt{rchi}) \wedge \exists x_1 \dots x_n \in \mathcal{V}\mathtt{ar}(\mathcal{A}\mathtt{rchi}) : P = [x_1, \dots, x_n, y]\}$$

où Δ_P *est le temps maximum de sortie du régime initial d'un chemin* P.

Justification. Pour l'image d'une observation, on choisit donc ce temps en fonction des chemins menant à cette variable. Lorsqu'un chemin sort du régime initial, l'horloge de ce chemin ne pointe plus vers l'occurrence initiale de la source. Il y a donc eu une mise à jour de l'image du chemin. Lorsqu'un chemin sort du régime initial, l'image aussi. On choisit donc comme temps de sortie du régime initial, le minimum des temps de sortie des chemins qui mènent à cette variable.

5.3.2 *Choix du temps de sortie du régime initial d'un chemin*

On choisit un temps de sortie du régime initial pour les chemins selon les propriétés de la spécification.

Choix 4. *Temps maximum de sortie du régime initial d'un chemin. Soit une spécification* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop} \rangle$. *Le temps maximum de sortie du régime initial* Δ_P *d'un chemin* P, *bien défini, d'une variable* x *à une variable* y *et sur lequel est spécifiée une propriété de décalage ou une propriété de latence avec une borne supérieure finie, est choisi égal à :*

$$\Delta_P = \Delta_x + \min(\mathtt{MaxLatency}(P), \mathtt{MaxShift}(P))$$

où Δ_x *est le temps maximum avant la première mise à jour de* x.

Justification. On examine les différents prédicats utilisés pour définir les propriétés d'un chemin de propagation afin de justifier pourquoi le temps maximum de sortie est défini par rapport au prédicat de latence et de décalage.

La propriété de latence limite le temps entre l'apparition d'une occurrence de la source et les instants d'utilisation de cette occurrence pour définir l'occurrence de l'image. Pour un chemin P, en tout instant en dehors du régime initial, l'image doit être définie par une occurrence de la source apparue au pire $MaxLatency(P)$ unités de temps auparavant. Lorsque la source est mise à jour et sort de son régime initial et ensuite lorsque $MaxLatency(P)$ unités de temps se sont écoulées, on choisit alors de forcer l'image a être définie par l'occurrence de sortie du régime initiale de la source ou une occurrence plus récente. On choisit donc d'utiliser ce temps $MaxLatency(P)$ et le temps de sortie du régime initial de la source pour définir le temps de sortie du régime initial de P.

La propriété de décalage, en dehors du régime initial, indique qu'en tout instant le décalage temporel associé au décalage logique ne doit pas excéder $MaxShift(P)$. Donc, pour une occurrence apparue en un instant t, en un instant $t + MaxShift(P)$, l'horloge de l'observation doit se référer à un état à un instant supérieur à t et donc vers une occurrence plus récente que celle apparue à t. On étend ce principe à la sortie du régime initial et on définit donc le temps de sortie du régime initial du chemin en fonction de $MaxShift(P)$ et du temps de sortie du régime initial de la source.

La propriété de lag détermine uniquement quand on peut mettre à jour l'image avec une occurrence de la source en limitant le décalage entre la mise à jour de la source et la mise à jour correspondante de l'image. Cependant, le lag n'impose pas de mettre à jour. Le lag n'intervient donc pas dans le choix du temps de sortie du régime initial d'un chemin.

Finalement, les propriétés de stabilité, de fraîcheur et de pérennité dépendent des états où l'occurrence de la source est prise par cette source et pas de l'instant actuel, de l'image ou du décalage le long du chemin. Ces propriétés n'interviennent donc pas dans le choix du temps de sortie du régime initial d'un chemin.

Choix 5. *Temps maximum de sortie du régime initial d'un chemin. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$. *Le temps maximum de sortie du régime initial* Δ_P *d'un chemin* P, *bien défini, d'une variable* x *à une variable* y *et sur lequel il n'y a pas de propriété de décalage ou de propriété de latence avec une borne supérieure finie spécifiée, est choisi égal à :*

$$\Delta_P = \min\{\Delta_{P'} | P' \in \mathcal{P}aths(\mathcal{A}rchi) \wedge \exists P_1, P_2 : P' = P_1 : P : P_2\}$$

où $\Delta_{P'}$ *est le temps maximum de sortie du régime initial d'un chemin* P'.

S'il n'y a pas de propriété de latence ou de décalage avec une borne supérieure finie spécifiée pour un chemin, alors le temps de sortie du régime initial de ce chemin est le minimum des temps de sortie du régime initial de ses sur-chemins.

Justification. Soit un chemin P pour lequel il n'y a pas de propriété de décalage ou de propriété de latence avec une borne supérieure finie spécifiée. On suppose qu'il existe un chemin $P' == [x_0, x_1, \ldots, x_n]$ tel que :

$$\exists P_1, P_2 : P' = P_1 : P : P_2$$

Alors P' est un sur-chemin de P et il existe deux entiers k, l tels que :

$$0 \leq k < l \leq n \wedge P = [x_k, x_{k+1}, \ldots, x_l]$$

Soit une exécution σ vérifiant la spécification et un état σ_i tel que P' est sorti de son régime initial. Alors $\forall c \in \mathcal{C}(P).\sigma : \hat{\hat{x}}_0.\sigma_{c(i)} \neq 0$. Soit trois horloges c', c'', c'' tel que :

$$c' \in \mathcal{C}([x_0, x_1, \dots x_k]).\sigma \wedge c'' \in \mathcal{C}([x_k, x_{k+1}, \dots x_l]).\sigma \wedge c'' \in \mathcal{C}([x_l, x_{l+1}, \dots x_n]).\sigma$$

Alors, d'après la proposition 22 page 63, on a $c' \circ c'' \in \mathcal{C}([x_0, x_1, \dots x_k]).\sigma$ et donc $c' \circ c'' \circ c''' \in \mathcal{C}(P).\sigma$. On a donc :

$$\hat{\hat{x}}_0.\sigma_{c' \circ c'' \circ c'''(i)} \neq 0$$

Dans l'état $\sigma_{c'' \circ c'''(i)}$, l'horloge c' du chemin $[x_0, x_1, \dots x_k]$ se réfère à un état et à une occurrence de x_0 sortie du régime initial, ce chemin est donc sorti du régime initial. Dans cet état, la variable x_l est donc aussi sortie du régime initial. Dans l'état $\sigma_{c'''(i)}$, de même, il existe une horloge c'' du chemin $P = [x_k, x_{k+1}, \dots x_l]$ qui se réfère à une occurrence de x_l sortie du régime initial, ce chemin P est alors sortie de son régime initial.

On en déduit que pour un chemin P, lorsqu'un de ses sur-chemins sort du régime initial alors ce chemin est aussi sorti du régime initial.

5.4 SORTIE DU RÉGIME INITIAL D'UNE SPÉCIFICATION EN TEMPS BORNÉ

Le temps de sortie du régime initial du système dépend du temps de sortie du régime initial des chemins, temps que l'on a choisi comme dépendant du temps de sortie du régime initial des sources des chemins. Or ce temps de sortie du régime initial des sources des chemins dépend lui-même du temps de sortie du régime initial des chemins. Il s'agit de trouver une condition suffisante sur le système qui ne limite pas l'expressivité du formalisme et permet de borner et calculer un temps de sortie du régime initial de la spécification et donc des chemins et variables de l'architecture.

5.4.1 *Définitions préliminaires*

On s'intéresse notamment aux variables qui ne sont pas images d'une observation. Pour ces variables, et si leur comportement est limité par une propriété de sporadicité ou de périodicité, le temps de sortie du régime initial n'est pas lié à un chemin et est borné. On veut une architecture où les occurrences de toutes les variables du système sont liées par un chemin aux occurrences de variables non image d'une observation. On s'intéresse aux architectures représentant un graphe ayant la propriété suivante :

Définition 55. *Graphe à entrées élémentaires Soit un graphe orienté* $G = (S, A)$. *Ce graphe est à entrées élémentaires si l'on a :*

$$\forall s \in S, \exists s' \in S : s' \rightarrow^* s \wedge \mathtt{prec}(s') = \varnothing$$

La notation $\rightarrow^*$ indique la présence d'un chemin entre deux sommets. $\mathtt{prec}(s)$ donne pour un sommet s ses prédécesseurs, c'est-à-dire l'ensemble des sommets pour lesquels il existe un arc vers s. On utilise par la suite la notation $\mathtt{succ}(s)$ qui définit l'ensemble des successeurs d'un sommet, c'est-à-dire l'ensemble des sommets pour lesquels il existe un arc de s vers ces sommets.

Pour une architecture dont le graphe vérifie cette propriété, pour chaque variable du système il existe un chemin de propagation depuis une variable non image d'une

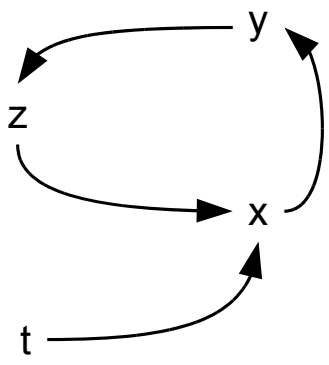

FIG. 16: Exemple de système à entrées élémentaires

observation. Dans le cas contraire, on peut notamment avoir un système sous forme de boucle, par exemple :

$$z \preccurlyeq y; y \preccurlyeq x; x \preccurlyeq z$$

Pour ce système, chacune des variables attend la sortie du régime initial des autres variables. On a donc un temps maximum de sortie du régime initial qui n'est pas borné. Un système à entrées élémentaires peut cependant avoir une boucle, par exemple le système illustrée figure 16 :

$$z \preccurlyeq y; y \preccurlyeq x; x \preccurlyeq f(z, t)$$

Ici, x sortira du régime initial grâce à t et les autres variables sortiront elles aussi du régime initial après x. Le système n'est alors pas fermé.

Pour caractériser un tel graphe, on donne les définitions suivantes :

Définition 56. *Graphe inversé. Soit un graphe orienté* $G = (S, A)$. *Le graphe inversé de* G *:* $G' = (S', A')$ *est définit par :*

$$S = S' \wedge (s \to s' \in A) \Leftrightarrow (s' \to s \in A')$$

Définition 57. *Composante fortement connexe. Soit un graphe orienté* $G = (S, A)$. *Une composante fortement connexe* C *de ce graphe est un sous-ensemble maximal de sommets tel qu'il existe un chemin entre tout couple de sommets de* C *:*

$$\forall s, s' \in C : s \to^* s' \wedge \forall s \in C, \forall s' \in S \setminus C : \neg(s \to^* s' \wedge s' \to^* s)$$

L'ensemble des composantes fortement connexes forme une partition du graphe.

Définition 58. *Composante fortement connexe terminale. Soit un graphe orienté* $G = (S, A)$. *Une composante fortement connexe* C *est terminale si on a :*

$$\forall s \in C : succ(s) \neq \varnothing \wedge succ(s) \subset C$$

Tout élément de C a alors au moins un successeur et tout ses successeurs sont dans C.

Proposition 32. *Un graphe à entrées élémentaires est un graphe dont aucune des composantes du graphe inversé fortement connexes est terminale.*

On démontre la proposition 32.

Démonstration. On prouve l'équivalence des définitions en prouvant que chaque définition implique l'autre.

1. On prouve tout d'abord qu'un graphe à entrées élémentaires est un graphe dont les composantes fortement connexes du graphe inversé ne sont pas fortement connexes terminales. Soit un graphe orienté $G = (S, A)$ à entrées élémentaires. Alors on a donc :

$$\forall s \in S, \exists s' \in S : s' \rightarrow^* s \ \wedge \ prec(s') = \varnothing$$

Pour le graphe inversé de G, $G' = (S', A')$, on a donc :

$$\forall s \in S, \exists s' \in S : s \rightarrow^* s' \ \wedge \ succ(s') = \varnothing$$

Soit C une composante fortement connexe de G' et s un sommet de C. Donc il existe $s' \in S$ tel que :

$$s \rightarrow^* s' \ \wedge \ succ(s') = \varnothing$$

De $succ(s') = \varnothing$ on déduit que s' n'est pas dans C. En effet, comme il n'y a pas de chemin partant de s', il ne peut y avoir de chemin de s' vers les autres éléments de C. On considère un chemin de s à s'. On prend alors le dernier sommet de ce chemin faisant parti de C (au pire ce sommet est celui juste avant s'). Alors le successeur de ce sommet n'est pas dans C et donc C n'est pas fortement connexe terminale.

2. On prouve désormais la réciproque. On considère un graphe $G = (S, A)$ dont les composantes fortement connexes du graphe inversé $G' = (S', A')$ ne sont pas fortement connexes terminales. Les composantes fortement connexes de G' forment une partition $C_1, C_2, C_3, \ldots, C_n$ de G'. On prouve que pour chacune de ces composantes connexes, soit elle est définie comme un singleton n'ayant pas de successeur, soit il existe un chemin de tous ses nœuds vers un nœud d'une autre composante connexe. On considère une des composantes connexes de G', C_1. Cette composante n'est pas fortement connexe terminale et donc on a :

$$\exists s \in C_1 : (succ(s) = \varnothing \ \vee \ succ(s) \notin C_1)$$

Si $succ(s)$ est vide, alors nécessairement $C_1 = \{s\}$, sinon il n'y a pas de chemin partant de s et le liant aux autres éléments de C_1 et donc C_1 ne serait pas fortement connexe.

Si $succ(s) \notin C_1$ alors il existe un successeur s' de s qui n'est pas dans C_1. s' fait nécessairement partie d'une autre composante fortement connexe, celles-ci formant une partition de G'. Comme pour tous les éléments de C_1 il existe un chemin vers s, alors il existe un chemin de tous les nœuds de C_1 vers s' en passant par s.

En ayant le même raisonnement pour chacune des composantes, on prouve donc que soit chaque composante est définie comme un singleton n'ayant pas de successeur, soit il existe un chemin de tous ses nœuds vers un nœud d'une autre composante connexe.

Lorsqu'il existe un chemin de tout les nœuds d'une composante connexe C vers une autre C', c'est que pour tous les nœuds de C, il existe un chemin vers tous les nœuds de C'. En effet, C' étant connexe, un chemin vers un de ses nœuds permet de construire

un chemin vers tous ses autres nœuds. L'inverse est faux sinon C et C' ne sont pas maximales.

On suppose désormais qu'aucune des composantes connexes de G' n'est un singleton. Chaque composante a alors un chemin vers une autre composante. Le nombre de composantes étant fini, il existe alors une ou plusieurs boucles entre les composantes. Une telle boucle permet de construire un chemin de n'importe quel nœud d'une composante vers n'importe quel nœud d'une autre composante de la boucle et inversement. L'ensemble des composantes connexes formant une boucle est donc un ensemble connexe. Ces composantes connexes ne sont alors pas maximales ce qui est en contradiction avec leur définition. Donc nécessairement il existe des composantes connexes de G' étant des singletons. De plus pour toute composante connexe de G', il existe des chemins vers une de ces composantes singletons sinon on a donc une boucle qui contredit la définition des composantes comme étant maximales.

On en déduit que pour tout nœud de G', il existe un chemin vers un nœud appartenant à une composante singleton et n'ayant pas de successeur. Et donc le graphe G est à entrées élémentaires. □

Pour une telle architecture, il faut de plus que les propriétés temporelles définissent des contraintes de temps permettant de borner le temps de propagation le long des chemins et donc le temps de sortie du régime initial de ces chemins. On définit un graphe pondéré dont les caractéristiques sont déduites des propriétés temporelles. On construit une fonction que l'on utilisera comme fonction de valuation des arcs de ce graphe. Ce graphe est ensuite utilisé pour calculer, quand c'est possible, le temps maximum de sortie de chaque variable et chemin de l'architecture.

Définition 59. *Valuation d'un chemin. Pour une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, on définit une fonction de valuation V pour chaque couple de variables de l'architecture :*

$$\forall x, y \in \mathcal{V}ar(\mathcal{A}rchi) :$$
$$V(x,y) = \min \left\{ \Delta \;\middle|\; \begin{array}{l} \exists P \in \mathcal{P}aths(\mathcal{A}rchi) : \\ \Delta = \min(MaxShift([x];P;[y]), MaxLatency([x];P;[y])) \end{array} \right\}$$

Le poids de chaque arc est donc défini par le minimum des bornes supérieures sur le décalage ou sur la latence des chemins entre x et y.

Définition 60. *Graphe de calcul du temps maximum de sortie du régime initial. Pour une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, on définit le graphe orienté et pondéré $G = (S, A, V)$ associé pour le calcul du temps maximum de sortie du régime initial des variables de la spécification par :*

- *il y a un sommet par variable plus une source du système : $S = \mathcal{V}ar(\mathcal{A}rchi) \cup \{src\}$ $(src \notin \mathcal{V}ar(\mathcal{A}rchi))$*
- *pour chaque chemin dont la valuation est bornée, il y a un arc pondéré par cette valuation de la source vers l'image :*

$$\forall x, y \in S \setminus \{src\} : x \overset{V(x,y)}{\rightarrow} y \; si \; V(x,y) \neq +\infty$$

- *pour chaque variable n'étant pas image d'une observation et pour laquelle il y a une propriété de sporadicité ou de périodicité, il y a un arc du sommet source du système vers le sommet de cette variable. Cet arc est pondéré par le temps maximum de sortie du régime initial de la variable déduit des propriétés de périodicité ou de sporadicité.*

$$src \overset{\Delta_x}{\rightarrow} x \; si \; x \in S, \nexists \, y, f, X : x \in X \wedge y \trianglelefteq f(X) \in \mathcal{A}rchi \wedge \Delta_x \neq +\infty$$

Définition 61. *Valuation d'un chemin. Soit un graphe orienté et pondéré* $G = (S, A, V)$*. Pour chaque chemin du graphe, on définit la valuation d'un chemin* $P = [x_1, x_2, \ldots, x_n]$ *par :*

$$V(P) = \sum_{i \in [1..n-1]} V(x_i, x_{i+1})$$

5.4.2 *Calcul du temps de sortie du régime initial*

En utilisant le graphe construit à partir d'une spécification, on a les propositions suivantes :

Proposition 33. *Soit une spécification* $\langle \mathcal{A}\mathrm{rchi}, \mathcal{P}\mathrm{rop} \rangle$ *et le graphe orienté et pondéré* $G = (S, A, V)$ *associé pour le calcul du temps maximum de sortie du régime initial des variables. Alors pour toute variable* x *et si dans le graphe* G *l' on a :*

$$\forall x \in S : src \rightarrow^* x$$

alors, d'après les choix 1, 2, 3, 4 sur les temps de sortie du régime initial, on a :

$$\Delta_x = \min\{V(P) | \exists x_1, \ldots, x_n \in S : P = [src, x_1, \ldots, x_n, x]\}$$

Le temps maximum de sortie du régime initial d'une variable est donc le plus court chemin entre la source du graphe et le nœud de cette variable.

Dans la preuve on distingue les chemins du graphe de calcul des chemins du graphe décrit par l'architecture. Par défaut on désigne par chemin, un chemin du graphe de calcul.

Démonstration. On donne cette preuve en quatre étapes :

1. on prouve tout d'abord que la valuation d'un chemin comportant une boucle est strictement positive ;
2. on prouve ensuite que le temps de sortie du régime initial d'un chemin P de l'architecture, dont les propriétés définissent la valuation d'un arc du graphe de calcul, s'il est borné, est défini par la valuation d'un chemin depuis la source vers le nœud représentant l'image de P ;
3. on prouve alors que le temps de sortie du régime initial d'une variable x, s'il est borné, est la valuation d'un chemin de la source vers le nœud de x ;
4. on prouve que le temps de sortie du régime initial d'une variable x, s'il est borné, est la valuation du plus court chemin de la source vers le nœud de x ;

1. On considère un chemin du graphe de calcul comportant une boucle. Une telle boucle existe s'il existe un chemin dans l'architecture de la spécification définissant une boucle passant par les mêmes variables. Les valuations des chemins le long de cette boucle sont déterminées par les prédicats de latence et de décalage des chemins de l'architecture. Dans le cadre d'une spécification sans boucle de causalité synchrone, il existe un sous-chemin de cette boucle ayant une propriété de décalage avec une borne minimum non nulle. Le minimum de la propriété de latence sur ce même chemin ne peut être nul sinon la proposition 29 donnant les liens entre les bornes minimum de la latence et du décalage n'est pas vérifiée. Les bornes maximum du décalage et de la latence devant être supérieur au borne minimum, la valuation de ce chemin est donc non nulle. La valuation le long de cette boucle est donc strictement positive.

2. On s'intéresse au temps de sortie du régime initial des chemins. Étant donné que le graphe ne donne qu'au plus un arc entre deux variables, on ne s'intéresse pour l'instant qu'aux chemins de l'architecture utilisés pour définir la valuation des arcs du graphe du temps de calcul. Soit un chemin P de l'architecture entre deux variables x et y et tel que :

$$V(x,y) = min(MaxShift([x];P;[y]), MaxLatency([x];P;[y]))$$

On a alors d'après le choix 4 :

$$\Delta_P = \Delta_x + V(x,y)$$

Soit un chemin P_1 de l'architecture d'une variable x_1 vers une variable x_0 tel que Δ_{P_1} est borné et égal à $\Delta_{x_1} + V(x_1,x_0)$. On prouve qu'il existe un nombre borné de variables $x_1, x_2, \ldots, x_n$ et où x_n n'est pas image d'une observation et n chemins $P_1, P_2, \ldots, P_n$ de l'architecture tel que :

$$\forall i \in [1..n] : V(x_i, x_{i-1}) = min(MaxShift(P_i), MaxLatency(P_i))$$

et tel que :

$$\Delta_{P_1} = \Delta_{x_n} + \sum_{i\in[1..n]} V(x_i, x_{i-1})$$

Pour Δ_{x_1}, il y a deux choix. Si jamais x_1 n'est pas image d'une observation, alors on a bien $\Delta_{P_1} = \Delta_{x_1} + V(x_1,x_0)$. Il n'y a alors pas d'autre variable, ni de chemin à considérer pour prouver la propriété.

Si x_1 est image d'une observation, alors il existe un chemin de l'architecture P_2 et une variable x_2 tel que :

$$\Delta_{x_1} = \Delta_{P_2} = \Delta_{x_2} + V(x_2,x_1)$$

C'est bien ce chemin P_2 de l'architecture qui est utilisé pour construire l'arc entre x_2 et x_1. Dans le cas contraire, on a un autre chemin entre x_2 et x_1 donnant une valuation plus petite et permettant de construire une valeur Δ_{x_1} plus petite et donc un temps de sortie du régime initial de x plus petit. Or d'après la définition 3, on s'intéresse au minimum. On a donc :

$$\Delta_{P_1} = \Delta_{x_2} + V(x_1,x_0) + V(x_2,x_1)$$

Par induction, on construit à chaque étape un chemin d'une variable x_k vers x_0 tel que Δ_{P_1} est égal à $\Delta_k + \sum_{i\in[1..k]} V(x_i, x_{i-1})$. On s'arrête lorsque la variable x_k n'est pas l'image d'une observation. Comme Δ_{P_1} est borné et que la valuation d'une boucle est strictement positive, on ne peut infiniment boucler. Il existe donc nécessairement une variable x_n, non image d'une observation et tel que :

$$\Delta_{P_1} = \Delta_{x_n} + \sum_{i\in[1..n]} V(x_i, x_{i-1})$$

Comme x_n n'est pas image d'une observation et Δ_{P_1} est borné, on a alors :

$$\Delta_{P_1} = V(src, x_n) + \sum_{i\in[1..n]} V(x_i, x_{i-1})$$

Le temps de sortie du régime initial d'un chemin P_1, dont les propriétés définissent la valuation d'un arc du graphe de calcul, est donc défini par la valuation d'un chemin de la source du graphe vers le nœud représentant l'image de P_1.

3. Pour une variable x_n, image d'une observation, on a Δ_{x_n} qui est le minimum des temps de sortie du régime initial des chemins dont x_n est l'image. Si Δ_{x_n} est borné alors il existe un chemin P de $x_n - 1$ à x_n tel que $\Delta_{x_n} = \Delta_P$ et Δ_P est borné. On prouve par l'absurde que ce sont les propriétés de ce chemin qui définissent la valuation de l'arc de x_{n-1} à x_n dans le graphe de calcul.

On suppose que les propriétés du chemin P ne sont pas celles définissant la valuation de l'arc de x_{n-1} à x_n. Il existe alors un autre chemin P' de x_{n-1} à x_n définissant $V(x_{n-1}, x_n)$. On a alors :

$$
\begin{array}{ll}
 & min(MaxShift(P), MaxLatency(P)) > min(MaxShift(P'), MaxLatency(P')) \\
\Rightarrow & \quad \{\text{ajout de } \Delta_{x_{n-1}}\} \\
 & \Delta_{x_{n-1}} + min(MaxShift(P), MaxLatency(P)) > \\
 & \Delta_{x_{n-1}} + min(MaxShift(P'), MaxLatency(P')) \\
\Rightarrow & \quad \{\text{choix 4}\} \\
 & \Delta_P > \Delta_{P'}
\end{array}
$$

Cette inégalité est contradictoire puisque P est défini comme le chemin dont le temps de sortie du régime initial définit le temps de sortie du régime initial de son image x_n. P est donc le chemin ayant le temps de sortie du régime initial minimum parmi les chemins de l'architecture ayant x_n comme image.

On en déduit donc que les propriétés du chemin P définissent la valuation de l'arc de x_{n-1} à x_n dans le graphe de calcul. D'après le deuxième point de la preuve, il existe alors un ensemble de variables $x_0, x_1, \ldots, x_{n-2}$ tel que :

$$\Delta_{x_n} = V(src, x_n) + \sum_{i \in [1..n]} V(x_i, x_{i-1})$$

Il existe un chemin de src vers y dont la valuation est égal à Δ_y.

4. On a prouvé que le temps de sortie du régime initial d'un chemin ou d'une variable est égal à la valuation d'un chemin depuis la source. On prouve désormais que c'est le plus court chemin de src à y qui définit Δ_y. On considère le plus court chemin de src à y. On suppose que ce chemin est défini par les nœuds $[src, x_1, x_2, \ldots, x_n, y]$. On prouve par récurrence sur ce chemin que pour tout $i \in [1..n]$, on a :

$$\Delta_{x_i} = V(src, x_1) + \sum_{j \in [2..n]} V(x_{j-1}, x_j)$$

Au rang 1, on sait qu'on a :

$$\Delta_{x_1} = V(src, x_1)$$

L'hypothèse de récurrence est donc vérifiée. On suppose que la propriété est vraie jusqu'à un rang k. On donc

$$\Delta_{x_k} = V(src, x_1) + \sum_{i \in [2..k]} V(x_{i-1}, x_i)$$

L'arc entre x_k et x_{k+1} représente un chemin de l'architecture P_{k+1} et on a alors :

$$\Delta_{P_{k+1}} = \Delta_{x_k} + V(x_k, x_{k+1})$$

Si jamais ce chemin P_{k+1} de l'architecture n'est pas celui qui définit $\Delta_{x_{k+1}}$ alors $\Delta_{x_{k+1}}$ est défini par la valuation d'un autre chemin de l'architecture P'_{k+1}, $\Delta_{x_{k+1}}$ étant choisi

comme étant le minimum des temps de sortie du régime initial des chemins de l'architecture menant à x_{k+1}.

Comme pour tout chemin, $\Delta_{P'_{k+1}}$ est égal à la valuation d'un chemin depuis la source src. Or ce chemin depuis la source ne peut être plus court que celui défini par les variables $x_1, x_2, \ldots, x_k$ sinon on peut construire avec ce chemin, un chemin plus court entre src et y. La valuation du chemin depuis la source qui est utilisée pour calculer $\Delta_{P'_{k+1}}$ définit donc un temps de sortie du régime initial de x_{k+1} plus élevé que $\Delta_{P_{k+1}}$. Ceci est contraire à la définition de P'_{k+1}. Donc c'est $\Delta_{P_{k+1}}$ et la valuation du chemin $[src, x_1, x_2, \ldots, x_k, x_{k+1}]$ qui définit $\Delta_{x_{k+1}}$.

En continuant la récurrence, on prouve que pour y, c'est bien le chemin le plus court depuis src qui définit Δ_y.

Pour toute variable x du système, si son temps de sortie du régime initial est borné, il est défini par le plus court chemin depuis la source du graphe de calcul vers le nœud correspondant à x. □

On donne maintenant les propriétés d'une spécification assurant que cette spécification va sortir du régime initial.

Proposition 34. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$. Alors le temps de sortie du régime initial de la spécification est borné et calculable si les trois conditions suivantes sont vérifiées :*

- *l'architecture définit un graphe à entrées élémentaires ;*
- *chaque chemin de l'architecture du système est soit soumis à un prédicat de latence ou de décalage avec une borne supérieure différente de $+\infty$, soit il existe un sur-chemin soumis à un prédicat de latence avec une borne supérieure différente de $+\infty$;*
- *chaque variable non image d'une observation est soumise à un prédicat de sporadicité ayant une borne supérieure ou est périodique.*

Démonstration. Il s'agit de démontrer que si ces trois conditions sont vérifiées, le temps de sortie du régime initial de chaque variable et chaque chemin peut être calculé en utilisant le graphe dédié à ce calcul. Si l'architecture du système définit un graphe à entrées élémentaires, c'est qu'il existe pour chaque variable du système un chemin de cette architecture d'une variable non image d'une observation vers cette variable.

La dernière condition permet de s'assurer que pour les variables non images d'une observation, il existe bien un arc de la source vers ces variables dans le graphe de calcul du temps de sortie du régime initial.

La condition sur les prédicats des chemins permet de s'assurer qu'il existe, pour chaque chemin variable de l'architecture image d'un chemin, un chemin depuis la source du graphe de calcul vers cette variable. Si jamais un chemin P est soumis à un prédicat de latence ou de décalage avec une borne supérieure différente de $+\infty$ alors les bornes sur ces prédicats définissent la valuation de l'arc du graphe de calcul allant de la source à l'image de ce chemin de l'architecture.

Une fois ces arcs construits, on suppose qu'il existe une variable y de l'architecture, image d'un chemin, et donc un nœud du graphe de calcul qui n'est pas relié à la source du graphe de calcul. Il existe dans l'architecture un chemin P allant d'une variable non-image d'une observation vers y. Ce chemin de l'architecture ne définit pas d'arc du graphe de calcul sinon y serait relié à la source de ce graphe. Cependant, il existe un sur-chemin de P soumis à un prédicat de latence avec une borne supérieure différente de $+\infty$. La proposition 30 permet de spécifier une borne supérieure sur la latence de P. Cette borne ne modifie pas la spécification et permet d'ajouter un arc avec une valuation bornée entre la source et l'image de P. On construit un tel arc si jamais il existe une variable pour laquelle il n'y a pas de chemins depuis la source.

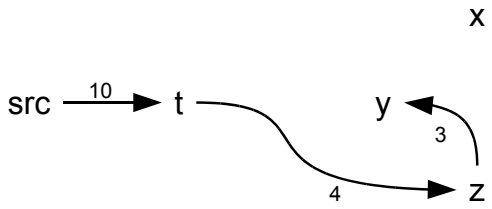

FIG. 17: Graphe de calcul de la spécification étendue

On en déduit donc que pour toute variable du système, on peut construire dans le graphe de calcul du temps de sortie du régime initial un chemin depuis la source. Les valuations de ces chemins définissent le temps de sortie du régime initial de ces variables. On en déduit le temps de sortie du régime initial de chacun des chemins de l'architecture en fonction des choix 4 et 5. □

Afin d'illustrer le calcul du temps de sortie du régime initial, on l'illustre dans le cas de l'architecture donnée figure 16.

$$z \lessdot y; y \lessdot x; x \lessdot f(z,t)$$

L'architecture définit bien un graphe à entrées indépendantes. On suppose que les propriétés temporelles de la spécification sont les suivantes :

$$t\ \{Sporadic(0,10)\}$$
$$[t,x,y,z]\ \{Latency(0,4)\}$$
$$[z,x,y]\ \{Shift(1,3)\}$$

La spécification vérifie la proposition 34 car l'entrée du système t est sporadique avec une borne supérieure et car tous les chemins du système sont soumis à une propriété de latence ou de décalage avec une borne supérieure ou alors un des sur-chemins est soumis à une propriété de latence avec une borne supérieure. On a donc un temps de sortie du régime initial borné. On construit à partir des propriétés temporelles le graphe de calcul de cette borne, graphe donné figure 17.

Sur ce graphe, il n'existe pas de chemin du nœud source src au nœud de la variable x, on ne peut donc pas utiliser ce graphe pour calculer le temps de sortie du régime initial de cette variable. On complète la spécification en utilisant la proposition 30 page 77 qui nous permet de déduire les propriétés suivantes à partir de la propriété de latence sur le chemin $[t,x,y,z]$:

$$[t,x]\ \{Latency(0,4)\}$$
$$[t,x,y]\ \{Latency(0,4)\}$$

Cela permet de compléter le graphe de calcul avec les deux arcs déduits de ces propriétés pour obtenir celui donné figure 18.

De ce graphe, on déduit les temps maximum de sortie de chacune des variables qui sont les longueurs des chemins les plus courts depuis le nœud source :

$$\Delta_t = 10$$
$$\Delta_x = 14$$
$$\Delta_y = 14$$
$$\Delta_z = 14$$

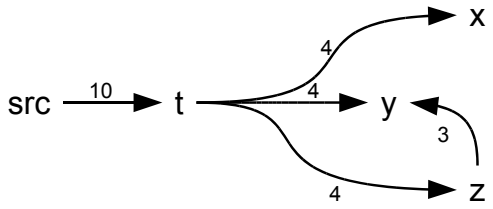

FIG. 18: Graphe de calcul de la spécification étendue

En ce qui concerne les chemins, les temps maximum de sortie du régime initial sont calculés selon le choix 4 :

$\Delta_{[t,x,y,z]} = 14$
$\Delta_{[z,x,y]} = 17$

Les chemins $[t, x, y, z]$ et $[z, x, y]$ sont des sur-chemins de tous les chemins du système ne comportant pas de boucles. En utilisant le choix 5, on déduit donc que les temps de sortie du régime initial des autres chemins de l'architecture sont bornés soit par 14, soit par 17.

Le temps maximum de sortie de la spécification est le maximum des temps de sortie du régime initial des variables et des chemins et est donc de 17.

6 SYSTÈME DE TRANSITIONS DÉFINI PAR UNE SPÉCIFICATION

Prouver la satisfiabilité d'une spécification, c'est prouver l'existence d'exécutions satisfaisant la spécification. Dans ce chapitre, on construit un modèle opérationnel équivalent à la spécification à base d'observation. Il ne s'agit pas d'une implantation mais un modèle opérationnel non déterministe. Ce modèle permet de construire état par état une exécution satisfaisant une spécification.

Ce modèle opérationnel est défini comme un système de transitions. La relation de transition de ce système est défini par un ensemble d'actions qui donnent les comportements possibles de chaque variable de l'architecture d'une spécification.

Dans ce chapitre, on introduit tout d'abord les différences entre la spécification donnée par un ensemble de relations d'observation et un système de transition. On motive le besoin de reformuler les propriétés de la spécification pour pouvoir définir à partir de ces propriétés une relation de transition, cette reformulation est donnée par la suite. En utilisant cette reformulation, on donne finalement la définition des actions définissant le système de transitions défini à partir des caractéristiques d'une spécification.

6.1 INTRODUCTION

6.1.1 *Satisfiabilité d'une spécification*

Dans certains cas, démontrer la non satisfiabilité d'une spécification est simple. On considère par exemple, une spécification décrivant la relation entre deux variables x et y et tel que $y \preccurlyeq x$. On suppose de plus que l'on a la propriété suivante sur le chemin allant de x à y :

$$[x,y]\,\{Lag(3,5), Latency(1,2)\}$$

Soit une exécution satisfaisant la spécification. On a alors :

$$\begin{array}{ll}
& \sigma \vDash [x,y]\,\{Lag(3,5), Latency(1,2)\} \\
\Rightarrow & \{\text{proposition 25}\} \\
& \sigma \vDash [x,y]\,\{Lag(3,5)\} \;\wedge\; \sigma \vDash [x,y]\,\{Latency(1,2)\} \\
\Rightarrow & \{\text{proposition 26}\} \\
& \sigma \vDash [x,y]\,\{Lag(3,5)\} \;\wedge\; \sigma \vDash [x,y]\,\{Latency(0,2)\} \\
\Rightarrow & \{\text{proposition 29}\} \\
& \sigma \vDash [x,y]\,\{Lag(3,5)\} \;\wedge\; \sigma \vDash [x,y]\,\{Lag(0,2)\} \\
\Rightarrow & \{\text{par définition en dehors du régime initial}\} \\
& \forall\, i \in \mathbb{N} : \left(\begin{array}{l} \exists c_1 \in \mathcal{C}([x,y]).\sigma : 3 \le \hat{\hat{y}}.\sigma_i - \hat{\hat{x}}.\sigma_{c_1(i)} < 5 \;\vee\; (T.\sigma_i < \Delta_{[x,y]} \;\wedge\; \hat{\hat{x}}.\sigma_{c_1(i)} = 0) \\ \qquad\qquad \wedge \\ \exists c_2 \in \mathcal{C}([x,y]).\sigma : 0 \le \hat{\hat{y}}.\sigma_i - \hat{\hat{x}}.\sigma_{c_2(i)} < 2 \;\vee\; (T.\sigma_i < \Delta_{[x,y]} \;\wedge\; \hat{\hat{x}}.\sigma_{c_2(i)} = 0) \end{array} \right) \\
\Rightarrow & \{\text{proposition 20}\} \\
& \forall\, i \in \mathbb{N} : \forall c \in \mathcal{C}([x,y]).\sigma : \\
& (3 \le \hat{\hat{y}}.\sigma_i - \hat{\hat{x}}.\sigma_{c_1(i)} < 5 \;\wedge\; 0 \le \hat{\hat{y}}.\sigma_i - \hat{\hat{x}}.\sigma_{c_1(i)} < 2) \;\vee\; (T.\sigma_i < \Delta_{[x,y]} \;\wedge\; \hat{\hat{x}}.\sigma_{c_1(i)} = 0) \\
\Rightarrow & \{\text{simplification}\} \\
& \forall\, i \in \mathbb{N} : \forall c \in \mathcal{C}([x,y]).\sigma : 3 < 2 \vee\; (T.\sigma_i < \Delta_{[x,y]} \;\wedge\; \hat{\hat{x}}.\sigma_{c_1(i)} = 0)
\end{array}$$

Ici, le chemin ne peut donc pas sortir du régime initial et donc si le temps de sortie de ce régime initial est borné, alors supposer qu'il existe une exécution vérifiant la spécification implique une contradiction. Par conséquent il n'existe pas d'exécution satisfaisant cette spécification et elle n'est donc pas satisfiable. On a construit une démonstration axiomatique de cette non satisfiabilité, démonstration simple étant donnée la simplicité de la spécification. Dans le cas de la spécification d'un système avec une architecture plus complexe et définissant un nombre plus important de propriétés temporelles, il est plus difficile de construire une telle démonstration de la satisfiabilité ou de la non satisfiabilité de cette spécification.

On cherche donc une autre méthode pour vérifier la satisfiabilité d'une spécification, méthode qu'il est important d'automatiser. C'est pour cela que l'on construit un modèle opérationnel défini comme un système de transition. Pour prouver la satisfiabilité d'une spécification, on utilise alors des algorithmes d'exploration des exécutions définies par un système de transitions.

6.1.2 *Architecture étudiée*

Dans ce chapitre on s'intéresse uniquement aux spécifications exprimées sous forme normale, sans boucle de causalité synchrone, dont l'architecture est satisfiable et dont le temps de sortie du régime initial est borné. La forme normale permet surtout de faciliter la manipulation des différentes propriétés de la spécification et des caractéristiques de ces propriétés. La satisfiabilité de l'architecture permet de définir le comportement du système uniquement par rapport aux propriétés temporelles et vérifier uniquement la satisfiabilité des propriétés temporelles. En effet, on s'intéresse au comportement temporel des variables et non aux valeurs effectivement prises par les variables. Finalement, on s'intéresse à une spécification dont le temps de sortie du régime initial est borné car on s'intéresse à des systèmes vivaces. On veut donc s'assurer que l'intégralité de l'architecture est vivace et parcourue par des occurrences autres que les occurrences initiales.

6.1.3 *Différence entre la spécification et un système de transitions*

Un système de transitions est défini par un triplet $(\Sigma, \Sigma_0, \rightarrow)$ ensemble d'états, ensemble d'états initiaux et relation de transition. La relation de transition définit à partir d'un état les possibles états successeurs. Dans le cas d'une spécification donnée sous la forme d'une architecture et de propriétés temporelles basées sur des relations d'observations, on a un ensemble de propriétés données sur la trace d'une exécution. De par les propriétés sur les chemins, l'état courant de l'image ne dépend pas uniquement de l'état antérieur.

En reformulant les propriétés de la spécification, on démontre qu'uniquement certaines caractéristiques des états antérieurs sont nécessaires à la définition des états postérieurs possibles. On introduit donc un ensemble de variables auxiliaires qui contiennent ces caractéristiques. En utilisant ces variables auxiliaires, on définit une relation de transition et donc un système de transitions dont on prouve que la sémantique est équivalente à celle d'un ensemble de relations d'observation et leurs propriétés temporelles.

6.1.4 *Action associée à une variable*

La relation de transition est définie comme un ensemble d'actions inspirées du langage TLA [Lam94] qui est présenté section 2.5.5. Chaque action est un prédicat entre deux états du système devant être vérifié pour qu'une transition entre ces deux états soit possible. Dans notre cas, afin de définir la relation de transition du système, on définit pour chaque variable du système une action décrivant le comportement de cette variable d'après les propriétés de la spécification.

On s'intéresse à la satisfiabilité des propriétés temporelles de la spécification. Ces propriétés définissent quand au cours de l'exécution les variables du système sont mises à jour et avec quelles occurrences des sources des chemins de propagation elles peuvent l'être. Le système de transitions équivalent à la spécification ne définit donc pas les valeurs prises par les variables de l'architecture mais les valeurs du profil temporel associé à chaque variable et le décalage le long de chaque chemin.

La relation de transition du système doit aussi définir le passage du temps. Dans le système défini par les relations d'observation, chaque évolution de la variable T correspond à l'écoulement d'une unité de temps. On sait de plus que le système est séparable et donc chaque modification d'une autre variable du système s'accompagne d'une modification de T. On définit alors la relation de transition du système comme la conjonction d'une action incrémentant T d'une unité de temps et des actions des variables de l'architecture.

Définition 62. *Relation de transition globale Étant donnée une spécification $\langle \mathcal{A}\mathrm{rchi}, \mathcal{P}\mathrm{rop}\rangle$ et l'ensemble des variables du système $\mathcal{V}\mathrm{ar}(\mathcal{A}\mathrm{rchi})) = \{x_k | k \in [1..n]\}$, on définit la relation de transition globale $\rightarrow$ de cette spécification par :*

$$\sigma_i \rightarrow \sigma_{i+1} \vDash T.\sigma_{i+1} = T.\sigma_i + 1 \ \wedge \ \bigwedge_{k=1}^{n} A_{x_k}.\sigma_i.\sigma_{i+1}$$

et où chaque A_{x_k} est une action, un prédicat portant sur deux états, définie à partir de la spécification et indiquant si la variable x_k peut être mise à jour et avec quelle occurrence. On appelle l'action A_{x_k} l'action de la variable x_k.

Lors de la définition des actions de chaque variable du système, on utilise la notation utilisée dans le langage TLA. Les variables primées d'une action donnent l'état des variables dans l'état suivant. Pour une action $A \triangleq x' = x + 1$, on a donc $A.\sigma_i.\sigma_{i+1} = (x.\sigma_{i+1} = x.\sigma_i + 1)$.

6.2 REFORMULATION DES PROPRIÉTÉS D'UNE SPÉCIFICATION

La relation de transition d'une variable définit si une variable est mise à jour et avec quelles occurrences des sources elle est mise à jour. On va chercher à exprimer en fonction du passé du système quelles sont les conditions nécessaires et suffisantes pour pouvoir mettre ou ne pas mettre à jour une variable. Ces conditions dépendent de deux éléments :

- les propriétés sur le comportement de la variable : la sporadicité et la périodicité ;
- les propriétés des chemins dont la variable est l'image ;

Les comportements autorisés par la relation de transition sont alors l'intersection des comportements autorisés par chaque propriété. Par exemple, la propriété de sporadicité peut autoriser la variable à être mise à jour ou à ne pas l'être alors que les propriétés sur

les chemins montrent qu'il n'existe pas d'occurrences valides disponibles pour mettre à jour cette variable. Dans ce cas, la variable ne devra pas être mise à jour.

Ces propriétés définissent le comportement autorisé des variables du système et l'évolution des liens entre ces variables au cours d'une exécution. On reformule chacune de ces propriétés sous la forme de conditions nécessaires et suffisantes pour que connaissant un état et ses prédécesseurs on puisse définir les différents états postérieurs autorisés par ces propriétés. Le but est de définir à partir de ces conditions une relation de transition définissant des exécutions satisfaisant ces propriétés.

6.2.1 *Sporadicité et périodicité*

Les propriétés de sporadicité et de périodicité définissent quand une variable peut ou doit être mise à jour indépendamment du reste du système. On reformule ces propriétés en commençant par la propriété de sporadicité. Pour que la sporadicité soit vérifiée dans un état, en fonction des états antérieurs, on a la proposition suivante :

Proposition 35. *Conditions nécessaires et suffisantes pour la sporadicité d'une variable. Soit une spécification* $\langle \mathcal{A}\mathrm{rchi}, \mathcal{P}\mathrm{rop}\rangle$*, une variable* x *et deux entiers* $\delta \in \mathbb{N}$ *et* $\Delta \in \mathbb{N} \cup \{+\infty\}$*. On a alors pour toute exécution* σ *vérifiant la spécification l'équivalence suivante :*

$$\forall\, i \in \mathbb{N} : (d_{\hat{x}}.\sigma_i \in [\delta..\Delta] \ \vee\ (T.\sigma_i < \Delta_x \ \wedge\ \hat{x}.\sigma_i = 0))$$
$$\Leftrightarrow$$
$$\forall\, i \in \mathbb{N} : T.\sigma_{i+1} \neq T.\sigma_i \Rightarrow \left(\begin{array}{c} (\hat{x}.\sigma_i \neq \hat{x}.\sigma_{i+1} \ \wedge\ \mathrm{SporadicUpdate}(T.\sigma_i, \hat{x}.\sigma_i, \delta, \Delta)) \\ \vee\ (\hat{x}.\sigma_i = \hat{x}.\sigma_{i+1} \ \wedge\ \mathrm{SporadicIdle}(T.\sigma_i, \hat{x}.\sigma_i, \delta, \Delta)) \end{array} \right)$$

avec :

$$\mathrm{SporadicUpdate}(T, \hat{x}, \delta, \Delta) \triangleq (\hat{x} \neq 0 \ \wedge\ \delta \leq T + 1 - \hat{x} \leq \Delta) \ \vee\ (\hat{x} = 0 \ \wedge\ T < \Delta_x)$$
$$\mathrm{SporadicIdle}(T, \hat{x}, \delta, \Delta) \triangleq (\hat{x} \neq 0 \ \wedge\ T + 1 - \hat{x} < \Delta) \ \vee\ (\hat{x} = 0 \ \wedge\ T + 1 < \Delta_x)$$

On donne une explication informelle de la proposition. On définit deux prédicats définissant les conditions soit pour mettre à jour, soit pour ne pas mettre à jour. La validité de ces prédicats se déduit des valeurs du temps T et du profil temporel $\hat{x}$ dans les états σ_i et σ_{i+1}. Dans le cas du régime initial, le choix est libre et il est possible de mettre à jour ou pas, une mise à jour entraînant une sortie de ce régime initial. En dehors du régime initial, s'il y a une mise à jour de x lors de la transition, c'est qu'on a nécessairement $\delta \leq T.\sigma_i + 1 - \hat{x}.\sigma_i \leq \Delta$. Si par contre il n'y a pas de mise à jour, c'est que l'on a nécessairement $T.\sigma_i + 1 - \hat{x}.\sigma_i < \Delta$. On en déduit que pour les instants tel que $\delta \leq T.\sigma_i + 1 - \hat{x}.\sigma_i < \Delta$, les deux transitions sont possibles. Dans les états tel que $T.\sigma_i + 1 - \hat{x}.\sigma_i < \delta$, on ne peut pas mettre à jour, il est encore trop tôt. Dans l'état tel que $T.\sigma_i + 1 - \hat{x}.\sigma_i = \Delta$, on ne peut que mettre à jour et la différence entre $T + 1$ et $\hat{x}$ ne peut donc jamais dépasser Δ.

Démonstration. On démontre l'équivalence des deux définitions en montrant que chacune des définitions implique l'autre.

1. Pour une variable x, si la spécification définit qu'une exécution valide vérifie la propriété suivante avec $\delta \in \mathbb{N}$ et $\Delta \in \mathbb{N} \cup \{+\infty\}$ alors on a :

$$\begin{array}{ll} & \sigma \vDash x\ \{\mathrm{Sporadic}(\delta, \Delta)\} \\ \Leftrightarrow & \quad \{\text{par définition}\} \\ & \forall\, i \in \mathbb{N} : d_{\hat{x}}.\sigma_i \in [\delta..\Delta] \ \vee\ (T.\sigma_i < \Delta_x \ \wedge\ \hat{x}.\sigma_i = 0) \end{array}$$

On suppose que l'on a une exécution vérifiant cette propriété et on démontre qu'on a alors :

$$\forall\, i \in \mathbb{N} : T.\sigma_{i+1} \neq T.\sigma_i \Rightarrow \left(\begin{array}{c} (\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_{i+1} \ \wedge\ \mathtt{SporadicUpdate}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, \delta, \Delta)) \\ \vee\ (\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{i+1} \ \wedge\ \mathtt{SporadicIdle}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, \delta, \Delta)) \end{array} \right)$$

Soit un entier i tel que $T.\sigma_{i+1} \neq T.\sigma_i$. On a alors $T.\sigma_{i+1} = T.\sigma_i + 1$. La propriété de sporadicité est vérifiée en tout état et donc dans les états σ_i et σ_{i+1}. On considère quatre cas possibles et on montre que dans chacun des cas, selon qu'il y a une mise à jour, soit le prédicat $\mathtt{SporadicUpdate}$ est vrai, soit le prédicat $\mathtt{SporadicIdle}$.

1. A. Si jamais dans l'état σ_{i+1}, x est dans le régime initial, alors il n'y a pas eu de mise à jour entre les deux états et on a donc $\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{i+1} = 0$. Le prédicat $\mathtt{SporadicIdle}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, \delta, \Delta)$ est alors vérifié car l'on a d'après la sporadicité en σ_{i+1} :

$$\begin{array}{ll} & T.\sigma_{i+1} < \Delta_x \ \wedge\ \hat{\hat{x}}.\sigma_{i+1} = 0 \\ \Leftrightarrow & \quad \{\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{i+1} \text{ et } T.\sigma_{i+1} = T.\sigma_i + 1\} \\ & T.\sigma_i + 1 < \Delta_x \ \wedge\ \hat{\hat{x}}.\sigma_i = 0 \end{array}$$

Alors l'implication est bien vérifiée.

1. B. On suppose désormais que l'état σ_{i+1} n'est plus dans le régime initial mais que dans l'état antérieur, x était dans le régime initial, alors on a nécessairement eu une mise à jour et $\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_{i+1}$. Le prédicat $\mathtt{SporadicUpdate}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, \delta, \Delta)$ est alors bien vérifiée car dans l'état σ_i, on a

$$\hat{\hat{x}}.\sigma_i = 0 \ \wedge\ T.\sigma_i < \Delta_x$$

1. C. On suppose que dans les deux états σ_i et σ_{i+1}, x n'est plus dans l'état initial. Alors on a nécessairement $\hat{\hat{x}}.\sigma_i \neq 0$. On suppose de plus que x est mis à jour entre σ_i et σ_{i+1}. Si elle est mise à jour, alors l'instant de mise à jour est lui aussi mis à jour et on a : $\hat{\hat{x}}.\sigma_{i+1} = T.\sigma_{i+1}$. On a alors :

$$\hat{\hat{x}}.\sigma_{i+1} = T.\sigma_{min\{j | j \geq i \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}}$$

En effet, dans l'état i, $T.\sigma_{min\{j | j \geq i \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}}$ est la date de la prochaine mise à jour. On en déduit que $T.\sigma_{min\{j | j \geq i \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} = T.\sigma_{i+1}$. On suppose que la sporadicité est vérifiée et donc on :

$$\delta \leq T.\sigma_{i+1} - \hat{\hat{x}}.\sigma_i \leq \Delta$$

Comme on sait que l'on a $T.\sigma_{i+1} = T.\sigma_i + 1$, on en déduit que l'on a :

$$\delta \leq T.\sigma_i + 1 - \hat{\hat{x}}.\sigma_i \leq \Delta$$

On a bien $\hat{\hat{x}}.\sigma_i \neq 0$ et le prédicat $\mathtt{SporadicUpdate}$ est vérifié.

1. D. Finalement, si dans les deux états σ_i et σ_{i+1}, x n'est plus dans l'état initial et s'il n'y a pas de mise à jour entre ces deux états. Alors on a $\hat{\hat{x}}.\sigma_{i+1} = \hat{\hat{x}}.\sigma_i$ car on ne met

pas non plus à jour l'instant de mise à jour. Comme la prochaine mise à jour a lieu dans un état futur, on a de plus :

$$
\begin{array}{ll}
& T.\sigma_{i+1} < T.\sigma_{min\{j|j\geq i \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} \\
\Rightarrow & \quad \{\text{ajout de } -\hat{\bar{x}}.\sigma_i\} \\
& T.\sigma_{i+1} - \hat{\bar{x}}.\sigma_i < T.\sigma_{min\{j|j\geq i \wedge\ \bar{x}.\sigma_i \neq \bar{x}.\sigma_j\}} - \hat{\bar{x}}.\sigma_i \\
\Rightarrow & \quad \{\text{sporadicité de } x\} \\
& T.\sigma_{i+1} - \hat{\bar{x}}.\sigma_i < \Delta \\
\Rightarrow & \quad \{\text{évolution de } T\} \\
& T.\sigma_i + 1 - \hat{\bar{x}}.\sigma_i < \Delta
\end{array}
$$

Et donc le prédicat $SporadicIdle$ est vérifié car on a $\hat{\bar{x}}.\sigma_i \neq 0$. On a prouvé la première implication.

2. On prouve désormais l'implication inverse. On suppose donc que l'on a :

$$
\forall\ i \in \mathbb{N} : T.\sigma_{i+1} \neq T.\sigma_i \Rightarrow \left(\begin{array}{c} (\hat{\bar{x}}.\sigma_i \neq \hat{\bar{x}}.\sigma_{i+1} \ \wedge\ SporadicUpdate(T.\sigma_i, \hat{\bar{x}}.\sigma_i, \delta, \Delta)) \\ \vee\ (\hat{\bar{x}}.\sigma_i = \hat{\bar{x}}.\sigma_{i+1} \ \wedge\ SporadicIdle(T.\sigma_i, \hat{\bar{x}}.\sigma_i, \delta, \Delta)) \end{array} \right)
$$

On veut prouver que pour tout entier i, on a :

$$
d_{\bar{x}}.\sigma_i \in [\delta..\Delta] \ \vee\ (T.\sigma_i < \Delta_x \ \wedge\ \hat{\bar{x}}.\sigma_i = 0)
$$

On s'intéresse uniquement aux transitions tel que le temps augmente. En effet, si jamais entre deux états σ_i et σ_{i+1} le temps n'augmente pas, alors d'après la séparabilité des exécutions du système, les variables n'évoluent pas et donc si la propriété de sporadicité est vérifiée en σ_i, elle l'est aussi en σ_{i+1}.

On fait une preuve occurrence par occurrence. Pour une valeur du compteur occ_x, on prouve que la propriété est vérifiée pour tous les états où le compteur a cette valeur occ_x. On distingue deux cas, en fonction du régime initial.

2. A. On considère tout d'abord la première occurrence et donc le régime initial dans lequel ce compteur est à son minimum. Pour tout ces états, on a alors la valeur du profil temporel $\hat{\bar{x}}$ qui est nul. On sort de ce régime initial dans un état σ_{i+1} tel que $\hat{\bar{x}}.\sigma_i \neq \hat{\bar{x}}.\sigma_{i+1}$. On prouve par récurrence que la propriété est vraie dans tout état entre 0 et $i+1$. Dans l'état σ_0, on a bien $T.\sigma_0 < \Delta_x$ et $\hat{\bar{x}}.\sigma_0 = 0$.

Ensuite, pour tous entier j tel que $j < i-1$, il n'y a pas de mise à jour et $\hat{\bar{x}}.\sigma_j = 0$ et donc on a :

$$
\begin{array}{ll}
& \hat{\bar{x}}.\sigma_j = \hat{\bar{x}}.\sigma_{j+1} \\
\Rightarrow & \quad \{SporadicIdle(T.\sigma_j, \hat{\bar{x}}.\sigma_j, \delta, \Delta)\} \\
& (\hat{\bar{x}}.\sigma_j \neq 0 \ \wedge\ T + 1 - \hat{\bar{x}}.\sigma_j < \Delta) \ \vee\ (\hat{\bar{x}}.\sigma_j = 0 \ \wedge\ T.\sigma_j + 1 < \Delta_x) \\
\Rightarrow & \quad \{\hat{\bar{x}}.\sigma_j = 0\} \\
& T.\sigma_j + 1 < \Delta_x \ \wedge\ \hat{\bar{x}}.\sigma_j = 0 \\
\Rightarrow & \quad \{T.\sigma_{j+1} = T.\sigma_j + 1 \text{ et } \hat{\bar{x}}.\sigma_j = \hat{\bar{x}}.\sigma_{j+1}\} \\
& T.\sigma_{j+1} < \Delta_x \ \wedge\ \hat{\bar{x}}.\sigma_{j+1} = 0
\end{array}
$$

La propriété de sporadicité est alors vérifiée dans tout état du régime initial.

2. B. On suppose que pour toutes les occurrences précédentes, la propriété est vraie. Soit un ensemble d'états ayant la même valeur du compteur d'occurrences et en dehors du régime initial. On a donc la valeur du profil temporel qui est non nul. On considère

l'ensemble des états où la valeur du compteur d'occurrences reste constante. Pour un tel état i et s'il n'y a pas de mise à jour de x lors de l'état suivant, alors on a :

$$
\begin{array}{ll}
& \hat{x}.\sigma_i = \hat{x}.\sigma_{i+1} \\
\Rightarrow & \quad \{SporadicIdle(T.\sigma_i, \hat{x}.\sigma_i, \delta, \Delta)\} \\
& (\hat{x}.\sigma_i \neq 0 \ \wedge \ T.\sigma_i + 1 - \hat{x} < \Delta) \ \vee \ (\hat{x}.\sigma_i = 0 \ \wedge \ T.\sigma_i + 1 < \Delta_x) \\
\Rightarrow & \quad \{\hat{x}.\sigma_i \neq 0\} \\
& T.\sigma_i + 1 - \hat{x} < \Delta \\
\Rightarrow & \quad \{T.\sigma_{i+1} = T.\sigma_i + 1\} \\
& T.\sigma_{i+1} - \hat{x} < \Delta
\end{array}
$$

Le temps étant vivace, cette inégalité ne peut être vraie tout au long de l'exécution. On a donc nécessairement un état j tel qu'il y aura une mise à jour de x terminant cette occurrence dans l'état suivant. On a alors :

$$
\begin{array}{ll}
& \hat{x}.\sigma_j \neq \hat{x}.\sigma_{j+1} \\
\Rightarrow & \quad \{SporadicUpdate(T.\sigma_j, \hat{x}.\sigma_j, \delta, \Delta)\} \\
& (\hat{x}.\sigma_j \neq 0 \ \wedge \ \delta \leq T.\sigma_j + 1 - \hat{x}.\sigma_j \leq \Delta) \ \vee \ (\hat{x}.\sigma_j = 0) \\
\Rightarrow & \quad \{\hat{x}.\sigma_j \neq 0\} \\
& \delta \leq T.\sigma_j + 1 - \hat{x}.\sigma_j \leq \Delta \\
\Rightarrow & \quad \{T.\sigma_{i+1} = T.\sigma_i + 1\} \\
& \delta \leq T.\sigma_{j+1} - \hat{x}.\sigma_j \leq \Delta
\end{array}
$$

Comme on a une mise à jour en σ_{j+1}, on a alors donc $d_{\bar{x}}.\sigma_j = T.\sigma_{j+1} - \hat{x}.\sigma_j$ et donc :

$$\delta \leq d_{\bar{x}}.\sigma_j \leq \Delta$$

La valeur de $d_{\bar{x}}$ étant la même tout le long de cette occurrence, chacun des états de cette occurrence vérifie la sporadicité. □

Dans le cas d'une variable périodique, on a le même raisonnement afin de déterminer dans quel cas prolonger l'occurrence courante et dans quel cas mettre à jour. On suppose donc que pour une variable x, la spécification définit qu'une exécution valide vérifie la propriété suivante :

$$
\begin{array}{l}
\sigma \vDash x \ \{Periodic(P, J, \phi)\} \triangleq \exists k \in \mathbb{N}, \exists \delta = \langle \delta_i | i \geq k \wedge \delta_i \in [-J..J] \rangle : \\
\{\hat{x}.\sigma_i | i \in \mathbb{N}\} = \{n * P + \phi + \delta_n | n \in \mathbb{N} \setminus [0..k-1]\} \cup \{0\} \wedge \ (k * P + \phi + \delta_n \leq \Delta_x)
\end{array}
$$

On a alors pour chaque entier $n \geq k$ une mise à jour et une seule aux instants autour de $n * P + \phi$ à la gigue J près. Il s'agit donc de s'assurer que les mises à jour se font uniquement dans les intervalles autorisées mais aussi de s'assurer qu'il n'y a qu'une mise à jour pour chaque intervalle. On doit donc vérifier s'il y a déjà eu une mise à jour dans le dernier intervalle où les mises à jour sont autorisées et l'effectuer ou non selon les cas. Un problème pourrait se poser si les intervalles possibles pour deux mises à jour successives se chevauchaient mais on s'intéresse uniquement au cas où la gigue J vérifie $J < P/2$ et donc ce cas n'est pas possible. On a alors la propriété suivante :

Proposition 36. *Conditions nécessaires et suffisantes pour la périodicité d'une variable. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$*, une variable* x *et trois entiers* $P, J, \phi \in \mathbb{N}$ *tel que* $J < P/2$*. On a alors pour toute exécution* σ *vérifiant la spécification l'équivalence suivante :*

$$\begin{array}{l} \exists k \in \mathbb{N}, \exists \delta = \langle \delta_i | i \geq k \wedge \delta_i \in [-J..J] \rangle : \\ \{\hat{x}.\sigma_i | i \in \mathbb{N}\} = \{n * P + \phi + \delta_n | n \in \mathbb{N} \setminus [0..k-1]\} \cup \{0\} \wedge \ (k * P + \phi + \delta_n \leq \Delta_x) \end{array}$$

$$\Leftrightarrow$$

$$\forall\, i \in \mathbb{N} : T.\sigma_i \neq T.\sigma_{i+1} : \left(\begin{array}{c} (\hat{x}.\sigma_i \neq \hat{x}.\sigma_{i+1} \wedge \mathtt{PeriodicUpdate}(T.\sigma_i, \hat{x}.\sigma_i, P, J, \phi)) \\ \vee \\ (\hat{x}.\sigma_i = \hat{x}.\sigma_{i+1} \wedge \mathtt{PeriodicIdle}(T.\sigma_i, \hat{x}.\sigma_i, P, J, \phi)) \end{array} \right)$$

où l'on a :

$$\begin{array}{l} \mathtt{PeriodicUpdate}(T, \hat{x}, P, J, \phi) \triangleq \textit{Let } n = \lfloor (T + 1 - \phi + J)/P \rfloor \textit{ in} \\ \quad (\hat{x} \neq 0 \ \wedge \ -J \leq T + 1 - n * P - \phi \leq J \ \wedge \ n - \lfloor (\hat{x} - \phi + J)/P \rfloor = 1) \\ \qquad \vee \\ (\hat{x} = 0 \ \wedge \ T < \Delta_x \ \wedge \ (\exists k \in \mathbb{N}, \exists \delta_k \in [-J..J] : T + 1 = k * P + \phi + \delta_k)) \end{array}$$

$$\begin{array}{l} \mathtt{PeriodicIdle}(T, \hat{x}, P, J, \phi) \triangleq \textit{Let } n = \lfloor (T + 1 - \phi + J)/P \rfloor \textit{ in} \\ (\hat{x} \neq 0 \ \wedge \ (T < n * P + \phi + J \ \vee \ \hat{x} \geq n * P + \phi - J)) \ \vee \ (\hat{x} = 0 \ \wedge \ T + 1 < \Delta_x) \end{array}$$

La notation $\lfloor r \rfloor$ renvoie à la partie entière de r. Donc $\lfloor (T + 1 - \phi + J)/P \rfloor$ donne l'entier n tel $n * P$ est le plus proche entier multiple de P inférieur à $T + 1 - \phi + J$. L'instant $n * P + \phi$ est donc l'instant le plus proche de T autour duquel une mise à jour a ou doit avoir lieu. Informellement, en dehors du régime initial, il ne peut y avoir mise à jour que dans les intervalles autorisées à chaque période et une seul fois. La condition pour ne pas mettre à jour est au contraire qu'on ne soit pas à la fin d'un intervalle où les mises à jour sont autorisées et sans qu'il n'y ait eu de mise à jour dans cet intervalle.

On donne la preuve de cette équivalence.

Démonstration. On démontre l'équivalence des deux définitions en montrant que chacune d'elles implique l'autre.

1. Pour une exécution σ où une variable x vérifie une propriété de périodicité, on a :

$$\begin{array}{ll} & \sigma \vDash x \ \{\mathtt{Periodic}(P, J, \phi)\} \\ \Leftrightarrow & \quad \{\text{par définition}\} \\ & \exists k \in \mathbb{N}, \exists \delta = \langle \delta_i | i \geq k \wedge \delta_i \in [-J..J] \rangle : \\ & \{\hat{x}.\sigma_i | i \in \mathbb{N}\} = \{n * P + \phi + \delta_n | n \in \mathbb{N} \setminus [0..k-1]\} \cup \{0\} \wedge \ (k * P + \phi + \delta_n \leq \Delta_x) \end{array}$$

On suppose que l'on a une telle exécution et on prouve que l'on a alors :

$$\forall\, i \in \mathbb{N} : \left(\begin{array}{c} (\hat{x}.\sigma_i \neq \hat{x}.\sigma_{i+1} \wedge \mathtt{PeriodicUpdate}(T.\sigma_i, \hat{x}.\sigma_i, P, J, \phi)) \\ \vee \\ (\hat{x}.\sigma_i = \hat{x}.\sigma_{i+1} \wedge \mathtt{PeriodicIdle}(T.\sigma_i, \hat{x}.\sigma_i, P, J, \phi)) \end{array} \right)$$

Soit un entier i tel que $T.\sigma_i \neq T.\sigma_{i+1}$, on a alors $T.\sigma_{i+1} = T.\sigma_i + 1$. On considère quatre cas possibles et on montre que dans chacun des cas, selon qu'il y a une mise à jour, soit le prédicat $\mathtt{PeriodicUpdate}$ est vrai, soit le prédicat $\mathtt{PeriodicIdle}$.

1. A. On suppose que dans l'état σ_{i+1} et donc aussi dans l'état σ_i, x est dans le régime initial. On a alors $\hat{\hat{x}}.\sigma_{i+1} = \hat{\hat{x}}.\sigma_i = 0$. Parmi les valeurs possibles de $\hat{\hat{x}}$, $k * P + \phi + \delta_k$ est la plus petite après 0. On sait que pour tout i : $T.\sigma_i \geq \hat{\hat{x}}.\sigma_i$, on a donc

$$T.\sigma_{i+1} \leq k * P + \phi + \delta_k \leq \Delta_x$$

Le seul cas d'égalité est quand $k * P + \phi + \delta_k = 0$ et comme $\Delta_x > 0$, on a bien $T.\sigma_{i+1} < \Delta_x$ et donc on a :

$$\hat{\hat{x}}.\sigma_i \neq 0 \ \wedge \ T.\sigma_i + 1 < \Delta_x$$

Le prédicat $\mathtt{PeriodicIdle}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)$ est vérifié.

1. B. On suppose désormais que dans l'état σ_{i+1}, x n'est plus dans le régime initial mais l'était dans l'état σ_i. On a donc une mise à jour. Étant donné que c'est la première mise à jour après le régime initial, on a donc $\hat{\hat{x}}.\sigma_{i+1} = k * P + \phi + \delta_k$. Dans l'état σ_i, x est dans le régime initial et donc on a vu que l'on a $T.\sigma_i < \Delta_x$. On a donc bien :

$$\hat{\hat{x}} = 0 \ \wedge \ T < \Delta_x \ \wedge \ (\exists k \in \mathbb{N}, \exists \delta_k \in [-J..J] : T = k * P + \phi + \delta)$$

Et donc le prédicat $\mathtt{PeriodicUpdate}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)$ est vérifié.

1. C. On suppose maintenant que dans les deux états σ_i et σ_{i+1}, x n'est plus dans le régime initial. On suppose de plus qu'il y a une mise à jour de x entre ces deux états. On a d'après les valeurs prises par le profil temporel x lors d'une exécution :

$$\exists k \in \mathbb{N}, \exists \delta_k, \delta_{k+1} \in [-J..J] : \hat{\hat{x}}.\sigma_i = k * P + \phi + \delta_k \ \wedge \ \hat{\hat{x}}.\sigma_{i+1} = (k+1) * P + \phi + \delta_{k+1}$$

On sait de plus que $\hat{\hat{x}}.\sigma_{i+1} = T.\sigma_{i+1} = T.\sigma_i + 1$. On a donc :

$$\lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor = \lfloor ((k+1) * P + \delta_{k+1} + J)/P \rfloor = k + 1$$

car on a $0 \leq \delta_{k+1} + J < P$. On a donc :

$$T.\sigma_i + 1 - \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P - \phi = \delta_{k+1} \in [-J..J]$$

et on a de plus :

$$\lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor - \lfloor (\hat{\hat{x}}.\sigma_i - \phi + J)/P \rfloor = 1$$

Cette appartenance à l'intervalle $[-J..J]$ et cette égalité prouvent donc que le prédicat $\mathtt{PeriodicUpdate}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)$ est alors vérifiée.

1. D. On suppose finalement que dans les deux états σ_i et σ_{i+1} n'est plus dans le régime initial et qu'il n'y a pas de mise à jour entre ces deux états. On prouve que $\mathtt{PeriodicIdle}((T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)$ est vrai et qu'on a :

$$\begin{array}{ll} & (T.\sigma_i < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J \ \vee \ \hat{\hat{x}}.\sigma_i \geq \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi - J) \\ \Leftrightarrow & \quad \{\text{reformulation}\} \\ & \neg(T.\sigma_i \geq \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J \ \wedge \ \hat{\hat{x}}.\sigma_i < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi - J) \end{array}$$

On suppose que l'on a :

$$(T.\sigma_i \geq \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J \ \wedge \ \hat{\hat{x}}.\sigma_i < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi - J)$$

et on montre que l'on a alors une contradiction. On a :

$$\exists k \in \mathbb{N}, \exists \delta_k \in [-J..J] : \hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{i+1} = k * P + \phi + \delta_k$$

Par hypothèse, on a alors :

$$\begin{array}{ll} & k * P + \phi + \delta_k < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi - J \\ \Rightarrow & \quad \{\delta_k \geq -J\} \\ & k * P + \phi - J < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi - J \\ \Rightarrow & \quad \{\text{simplification}\} \\ & k < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & k + 1 \leq \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor \end{array}$$

Or on a :

$$\begin{array}{ll} & T.\sigma_i \geq \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J \\ \Rightarrow & \quad \{k + 1 \leq \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor\} \\ & T.\sigma_i \geq (k + 1) * P + \phi + J \\ \Rightarrow & \quad \{T.\sigma_{i+1} > T.\sigma_i\} \\ & T.\sigma_{i+1} > (k + 1) * P + \phi + J \end{array}$$

Il existe $\delta_{k+1} \in [-J..J]$ tel que la prochaine date de mise à jour est $(k + 1) * P + \phi + \delta_{k+1}$. Comme on n'a pas encore eu cette mise à jour, on a donc :

$$\begin{array}{ll} & T.\sigma_{i+1} \leq (k + 1) * P + \phi + \delta_{k+1} \leq (k + 1) * P + \phi + J \\ \Rightarrow & \quad \{T.\sigma_{i+1} > (k + 1) * P + \phi + J\} \\ & (k + 1) * P + \phi + J < (k + 1) * P + \phi + J \\ \Rightarrow & \quad \{\text{simplification}\} \\ & 0 < 0 \end{array}$$

On a donc une contradiction et l'hypothèse de départ est fausse. On en déduit que le prédicat $\texttt{PeriodicIdle}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)$ est vérifié.

2. On prouve désormais la réciproque, on suppose que l'on a :

$$\forall\, i \in \mathbb{N} : \left(\begin{array}{c} (\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_{i+1} \wedge \texttt{PeriodicUpdate}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)) \\ \vee \\ (\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{i+1} \wedge \texttt{PeriodicIdle}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)) \end{array} \right)$$

et on montre que le profil temporel d'une variable x vérifiant cette propriété vérifie alors :

$$\begin{array}{l} \exists k \in \mathbb{N}, \exists \delta = \langle \delta_i | i \geq k \wedge \delta_i \in [-J..J] \rangle : \\ \{\hat{\hat{x}}.\sigma_i | i \in \mathbb{N}\} = \{n * P + \phi + \delta_n | n \in \mathbb{N} \setminus [0..k-1]\} \cup \{0\} \wedge\ (k * P + \phi + \delta_n \leq \Delta_x) \end{array}$$

Comme pour la sporadicité, on considère les différentes occurrences de x les unes après les autres. On effectue une récurrence sur ces occurrences en prouvant que pour chaque occurrence il existe deux entiers k et $\delta_k \in [-J..J]$ tel que cette occurrence finit par une mise à jour en un instant de la forme $k * P + \phi + \delta_k$.

2.A. On commence par la première occurrence au régime initial. Cette occurrence finit par une mise à jour et donc dans un état σ_{i+1} tel que $\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_{i+1}$. On a donc le prédicat $\mathtt{PeriodicUpdate}(T.\sigma_i, \hat{\hat{x}}.\sigma_i, P, J, \phi)$ et donc comme $\hat{\hat{x}}.\sigma_i = 0$:

$$T.\sigma_i < \Delta_x \ \wedge \ (\exists k \in \mathbb{N}, \exists \delta_k \in [-J..J] : T.\sigma_i + 1 = k * P + \phi + \delta_k)$$

Comme on a $\hat{\hat{x}}.\sigma_{i+1} = T.\sigma_i + 1$, on en déduit donc que cette occurrence finit bien dans un instant de la forme $k * P + \phi + \delta_k$ et que de plus on a $k * P + \phi + \delta_k \leq \Delta_x$.

Étant donné qu'on a $T.\sigma_i \geq 0$, on a donc $k * P + \phi + \delta_k > 0$ et donc si $k = 0 : \delta_k > -\phi$. En prenant une inégalité souple, on inclut la valeur du profil temporel dans l'état initial, c'est-à-dire 0.

2.B. On suppose désormais que toutes les occurrences précédentes finissent en un instant de la forme $k * P + \phi + \delta_k$. On s'intéresse donc à une occurrence qui commence en un instant de cette forme et en dehors du régime initial. On prend deux états de cette occurrence σ_i et σ_{i+1} et entre lesquels il n'y a donc pas de mise à jour. On a alors le prédicat $\mathtt{PeriodicIdle}$ en dehors du régime initial et donc :

$$(T.\sigma_i < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J \ \vee \ \hat{\hat{x}}.\sigma_i \geq n * P + \phi - J)$$

Si on considère un état tel que $T.\sigma_i + 1 = (k+1) * P + \phi + J$, on a alors :

$$\lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J = (k+1) * P + \phi + J$$

alors l'inégalité :

$$T.\sigma_i < \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P + \phi + J$$

n'est plus respectée. Le temps étant croissant, un tel état arrivera nécessairement et le prédicat $\mathtt{PeriodicIdle}$ ne sera plus vrai. On finit donc par avoir une mise à jour.

2.C. On prouve désormais que cette mise à jour est bien de la forme $k * P + \phi + \delta_k$.

Soit un état de cette occurrence σ_i tel qu'il y a une mise à jour dans le prochain état σ_{i+1} et donc $T.\sigma_{i+1} = \hat{\hat{x}}.\sigma_{i+1}$. Alors d'après le prédicat $\mathtt{PeriodicUpdate}$ et en dehors du régime initial on a :

$$(-J \leq T.\sigma_i + 1 - \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor * P - \phi \leq J \wedge \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor - \lfloor (\hat{\hat{x}}.\sigma_i - \phi + J)/P \rfloor = 1)$$

Il existe un entier l égal à $\lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor$. On a alors :

$$\begin{array}{ll} & \left(\begin{array}{c} -J \leq T.\sigma_i + 1 - l * P - \phi \leq J \\ \wedge \\ \lfloor (T.\sigma_i + 1 - \phi + J)/P \rfloor - \lfloor (\hat{\hat{x}}.\sigma_i - \phi + J)/P \rfloor = 1 \end{array} \right) \\ \Rightarrow & \{\hat{\hat{x}}.\sigma_i = k * P + \phi + \delta_k \text{ et } T.\sigma_i + 1 = T.\sigma_{i+1} = \hat{\hat{x}}.\sigma_{i+1}\} \\ & \left(\begin{array}{c} -J \leq \hat{\hat{x}}.\sigma_{i+1} - l * P - \phi \leq J \\ \wedge \\ l - k = 1 \end{array} \right) \\ \Rightarrow & \{\text{réécriture}\} \\ & \left(\begin{array}{c} l * P + \phi - J \leq \hat{\hat{x}}.\sigma_{i+1} \leq l * P + \phi - J \\ \wedge \\ l = k + 1 \end{array} \right) \end{array}$$

On a donc bien un entier δ_{k+1} tel que :

$$\hat{\hat{x}}.\sigma_{i+1} = (k+1) * P + \phi + \delta_{k+1}$$

On en déduit que toutes les occurrences finissent par une mise à jour à un instant de cette forme. On a donc bien les valeurs du profil temporel de x lors de l'exécution qui vérifient la périodicité. □

6.2.2 *Propriétés sur la propagation des occurrences*

La mise à jour d'une variable correspond au début d'une nouvelle occurrence de cette variable. Si cette variable est l'image d'une observation, elle doit alors être mise à jour en utilisant de nouvelles occurrences des sources. Pour pouvoir mettre à jour une telle variable, il faut donc respecter les propriétés sur son comportement temporel mais aussi s'assurer de la disponibilité de nouvelles occurrences qui plus est temporellement valides. Si jamais on ne met pas à jour une variable, il faut aussi vérifier que les occurrences utilisées pour définir l'occurrence actuelle sont toujours temporellement valides.

Pour un chemin de propagation d'une variable x à une variable y, on prouve que le respect des propriétés temporelles du chemin le long d'une exécution est équivalent à l'utilisation en chaque état d'une occurrence de la source vérifiant certaines propriétés. On commence par démontrer la proposition suivante :

Proposition 37. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et un chemin P de la variable x vers la variable y bien défini par $\mathcal{A}rchi$. On démontre l'équivalence suivante :*

$$\exists c \in \mathcal{C}(P).\sigma, \forall i \in \mathbb{N} : \begin{pmatrix} Predicate_1(P, c, \delta_1, \Delta_1).\sigma_i \\ \wedge \ \ldots \ \wedge \\ Predicate_n(P, c, \delta_n, \Delta_n).\sigma_i \end{pmatrix}$$
$$\Leftrightarrow$$
$$\forall i \in \mathbb{N}^*, \exists c \in \mathcal{C}(P).\sigma : \begin{pmatrix} Predicate_1(P, c, \delta_1, \Delta_1).\sigma_i \\ \wedge \ \ldots \ \wedge \\ Predicate_n(P, c, \delta_n, \Delta_n).\sigma_i \end{pmatrix}$$

et où

$$\forall\ i \in [1..n] : Predicate_i \in \{Latency, Lag, Shift, Freshness, Fitness, Stability\}$$

Il s'agit de prouver que la construction d'une horloge vérifiant les propriétés d'un chemin le long d'une exécution peut se faire état par état. Cette construction est possible car les prédicats bornent l'écart entre l'instant actuel et l'instant observé. Lorsque la valeur de T croît, la date de l'instant observé doit donc croître. Si dans deux états successeurs, il existe des horloges vérifiant les prédicats portant sur les chemins, il est alors possible d'utiliser les valeurs de ces horloges pour construire une horloge qui vérifie les prédicats et qui est croissante entre ces deux états. Par récurrence, on peut alors construire une horloge croissante vérifiant les prédicats tout le long d'une exécution.

Démonstration. L'implication de haut en bas est due aux propriétés de la logique des prédicats et à la croissance des horloges et du profil temporel.

On s'intéresse à l'implication réciproque. Pour cela, on suppose que l'on a :

$$\forall i \in \mathbb{N}, \exists c \in \mathcal{C}(P).\sigma : \begin{pmatrix} Predicate_1(P, c, \delta_1, \Delta_1).\sigma_i \\ \wedge \ \ldots \ \wedge \\ Predicate_n(P, c, \delta_n, \Delta_n).\sigma_i \end{pmatrix}$$

On prouve alors que l'on peut construire une horloge qui vérifie ces prédicats tout le long de l'exécution. La démonstration n'est pas directe car si l'on construit une horloge avec les valeurs des différentes horloges vérifiant les prédicats dans chacun des états, on ne construit pas nécessairement une horloge croissante.

On utilise un raisonnement par récurrence sur les états le long d'une exécution. Dans l'état initial, on est dans le régime initial du chemin et tous les prédicats sont vérifiées. On suppose que l'on a un entier i et une horloge c tel que pour tout entier j inférieur ou égal à i, l'horloge vérifie les prédicats dans l'état σ_j et dans chaque état la valeur de c est la valeur minimum tel que les prédicats soient vérifiées. On s'intéresse alors à l'état σ_{i+1}. On considère quatre cas différents selon le régime initial.

1. Le premier cas est lorsque l'on reste dans le régime initial lors du passage de l'état σ_i à σ_{i+1} . On choisit alors $c(i+1) = c(i)$. Les propriétés du régime initial sont toujours vérifiées.

2. Le second cas est lorsque que l'on quitte le régime initial. On sait que l'on a :

$$\exists c' \in \mathcal{C}(P).\sigma : \left(\begin{array}{c} Predicate_1(P, c', \delta_1, \Delta_1).\sigma_{i+1} \\ \wedge \ \ldots \ \wedge \\ Predicate_n(P, c', \delta_n, \Delta_n).\sigma_{i+1} \end{array} \right)$$

On choisit alors pour $c(i+1)$ la valeur minimum parmi les valeurs prises par les horloges c' vérifiant les propriétés. On a nécessairement c croissante étant donné qu'on pointe vers une nouvelle occurrence de la source du chemin et donc vers des états futurs à l'état pointé par c dans l'état i. On a de plus $\hat{\bar{x}}.\sigma_{c(i+1)} > \hat{\bar{x}}.\sigma_{c(i)}$.

3. Pour les deux derniers cas, le chemin n'est plus dans le régime initial dans les deux états σ_i et σ_{i+1}. On suppose tout d'abord qu'il y a en σ_{i+1} une mise à jour de l'image du chemin avec une nouvelle occurrence de la source du chemin. De nouveau, on choisit pour $c(i+1)$ la valeur minimum parmi les valeurs prises par les horloges c vérifiant les propriétés. La croissance de c est assurée par le fait de pointer vers une occurrence plus récente de la source.

4. Finalement, on suppose que le chemin n'est plus dans le régime initial, qu'il n'y a pas eu de mise à jour de l'image et donc que dans les deux états, c'est la même occurrence de la source qui est utilisée pour définir l'occurrence de l'image. On suppose aussi que la valeur de l'horloge en i est la plus petite vérifiant les propriétés dans l'état σ_i. On prouve alors que les valeurs possibles des horloges vérifiant les propriétés dans l'état σ_{i+1} sont nécessairement supérieures ou égales.

Pour les prédicats de lag, de latence et de stabilité, le respect de ces prédicats ne dépend pas de l'horloge mais uniquement des caractéristiques de l'occurrence source qui est utilisée. On étudie donc les prédicats de décalage, de fraîcheur et de pérennité. En dehors du régime initial, il existe donc des horloges c' tel que dans l'état σ_{i+1}, on a :

$$\delta_{Shi} \leq T.\sigma_{i+1} - T.\sigma_{c'(i+1)} < \Delta_{Shi} \ \wedge \ \delta_{Fit} < \hat{\bar{x}}.\sigma_{c'(i+1)} + d_{\bar{x}}.\sigma_{c'(i+1)} - T.\sigma_{c'(i+1)} \leq \Delta_{Fit}$$
$$\wedge \ \delta_{Fre} \leq T.\sigma_{c'(i+1)} - \hat{\bar{x}}.\sigma_{c'(i+1)} < \Delta_{Fre}$$

Il n'y a pas de mise à jour entre σ_i et σ_{i+1}, on a donc $\hat{\hat{x}}.\sigma_{c(i)} = \hat{\hat{x}}.\sigma_{c'(i+1)}$. On a de plus $T.\sigma_{i+1} = T.\sigma_i + 1$. Donc :

$$
\begin{array}{ll}
& \delta_{Shi} \leq T.\sigma_i + 1 - T.\sigma_{c'(i+1)} < \Delta_{Shi} \wedge \delta_{Fit} < \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - T.\sigma_{c'(i+1)} \leq \Delta_{Fit} \\
& \wedge \delta_{Fre} \leq T.\sigma_{c'(i+1)} - \hat{\hat{x}}.\sigma_{c(i)} < \Delta_{Fre} \\
\Rightarrow & \quad \{\text{réécriture}\} \\
& -\delta_{Shi} \geq T.\sigma_{c'(i+1)} - T.\sigma_i - 1 > -\Delta_{Shi} \wedge -\delta_{Fit} > T.\sigma_{c'(i+1)} - \hat{\hat{x}}.\sigma_{c(i)} - d_{\bar{x}}.\sigma_{c(i)} \geq -\Delta_{Fit} \\
& \wedge \delta_{Fre} \leq T.\sigma_{c'(i+1)} - \hat{\hat{x}}.\sigma_{c(i)} < \Delta_{Fre} \\
\Rightarrow & \quad \{\text{réécriture}\} \\
& T.\sigma_i + 1 - \delta_{Shi} \geq T.\sigma_{c'(i+1)} > T.\sigma_i + 1 - \Delta_{Shi} \\
& \wedge \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - \delta_{Fit} > T.\sigma_{c'(i+1)} \geq \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - \Delta_{Fit} \\
& \wedge \delta_{Fre} + \hat{\hat{x}}.\sigma_{c(i)} \leq T.\sigma_{c'(i+1)} < \Delta_{Fre} + \hat{\hat{x}}.\sigma_{c(i)}
\end{array}
$$

Entre l'état σ_i et l'état σ_{i+1} seules les conditions sur le décalage ont changé.

On prouve qu'on a nécessairement $c'(i+1) \geq c(i)$ par l'absurde. On suppose donc qu'il existe une horloge tel que $c'(i+1) < c(i)$. On a alors $T.\sigma_{c'(i+1)} < T.\sigma_{c(i)}$. On s'intéresse uniquement à la propriété de décalage. On a alors deux possibilités.

Si jamais on a aussi $\delta_{Shi} \leq T.\sigma_i - T.\sigma_{c'(i+1)} < \Delta_{Shi}$ alors c' vérifiait aussi le décalage dans l'état antérieur. Comme les conditions sur la fraîcheur et la pérennité sont les mêmes dans les deux états, c' vérifiait aussi ces propriétés dans l'état σ_{i+1}. On a alors une contradiction car en choisissant la plus petite valeur, on aurait pris au plus celle prise par c'.

On n'a donc pas $\delta_{Shi} \leq T.\sigma_i - T.\sigma_{c'(i+1)} < \Delta_{Shi}$. Étant donné que l'on a $T.\sigma_{c'(i+1)} < T.\sigma_{c(i)}$, on en déduit donc que $T.\sigma_{c'(i+1)}$ est plus petite que la borne inférieure des valeurs de $T.\sigma_{c(i)}$ vérifiant le décalage :

$$T.\sigma_{c'(i+1)} \leq T.\sigma_i - \Delta_{Shi}$$

Donc on a nécessairement :

$$
\begin{array}{ll}
& T.\sigma_{c'(i+1)} \leq T.\sigma_i + 1 - \Delta_{Shi} \\
\Rightarrow & \quad \{\text{réécriture}\} \\
& \Delta_{Shi} \leq T.\sigma_{i+1} - T.\sigma_{c'(i+1)}
\end{array}
$$

Et alors $c'(i+1)$ ne vérifie pas la propriété de décalage. On a donc nécessairement $c'(i+1) \geq c(i)$. On prend alors pour $c(i+1)$ la plus petite valeur des horloges c' vérifiant les propriétés en σ_{i+1}. On a $c(i) \leq c(i+1)$ et donc on construit bien une horloge croissante. □

Dans chaque état, il suffit de trouver une horloge du chemin vérifiant les propriétés du chemin en cet état pour être capable de construire une horloge vérifiant les propriétés du chemin tout au long de l'exécution. On donne désormais les conditions d'existence d'une telle horloge dans chaque état.

Proposition 38. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et un chemin P de la variable x vers la variable y bien défini par $\mathcal{A}rchi$. On suppose que l'on a un ensemble de paramètres S_Δ tel que :*

$$\delta_{Lat}, \delta_{Lag}, \delta_{Shi}, \delta_{Fre}, \delta_{Fit}, \delta_{Sta} \in \mathbb{N} \wedge \Delta_{Lat}, \Delta_{Lag}, \Delta_{Shi}, \Delta_{Fre}, \Delta_{Fit}, \Delta_{Sta} \in \mathbb{N} \cup \{+\infty\}$$

Soit une exécution σ vérifiant la spécification. On suppose que l'on a une occurrence de la source x du chemin P apparue à l'instant upd et conservée pendant dur instants sur la source. Pour tout entier i, on a alors l'équivalence suivante :

$$\left(\begin{array}{l}\exists c \in \mathcal{C}(\mathbb{N}) : \\ \hat{\hat{x}}.\sigma_{c(i+1)} = upd \wedge \left(\begin{array}{c} Latency(P,c,\delta_{Lat},\Delta_{Lat}).\sigma_i \wedge Lag(P,c,\delta_{Lag},\Delta_{Lag}).\sigma_i \\ \wedge Shift(P,c,\delta_{Shi},\Delta_{Shi}).\sigma_i \wedge Freshness(P,c,\delta_{Fre},\Delta_{Fre}).\sigma_i \\ \wedge Fitness(P,c,\delta_{Fit},\Delta_{Fit}).\sigma_i \wedge Stability(P,c,\delta_{Sta},\Delta_{Sta}).\sigma_i \end{array}\right)\end{array}\right)$$

$$\Leftrightarrow$$

$$\left(\begin{array}{c} upd = 0 \wedge T.\sigma_{i+1} < \Delta_P \\ \vee \\ upd \neq 0 \wedge dur \in DurationInterval(T.\sigma_{i+1}, upd, S_\Delta) \wedge \\ \left(\begin{array}{c} \hat{\hat{y}}.\sigma_{i+1} \neq \hat{\hat{y}}.\sigma_i \wedge T.\sigma_{i+1} \in PathInterval(upd, S_\Delta) \cap [upd+\delta_{Lag}..upd+\Delta_{Lag}[\\ \vee \\ \hat{\hat{y}}.\sigma_{i+1} = \hat{\hat{y}}.\sigma_i \wedge T.\sigma_{i+1} \in PathInterval(upd, S_\Delta) \end{array}\right) \end{array}\right)$$

avec :

$$PathInterval(upd, S_\Delta) \triangleq [max\left(\begin{array}{c} upd+\delta_{Lat} \\ upd+\delta_{Fre}+\delta_{Shi} \end{array}\right)..min\left(\begin{array}{c} upd+\Delta_{Lat}, \\ upd+\Delta_{Fre}+\Delta_{Shi}-1 \end{array}\right)[$$

et

$$DurationInterval(T, upd, S_\Delta) \triangleq [max\left(\begin{array}{c} \delta_{Sta}, \delta_{Fre}+\delta_{Fit}+1, \\ T-upd+\delta_{Fit}-\Delta_{Shi}+2 \end{array}\right)..min\left(\begin{array}{c} \Delta_{Sta}-1, \Delta_{Fre}+\Delta_{Fit}-1, \\ T-upd+\Delta_{Fit}-\delta_{Shi} \end{array}\right)]$$

Cette proposition donne donc les conditions nécessaires et suffisantes pour que l'utilisation d'une occurrence de la source dans l'état suivant permette de construire une horloge vérifiant les propriétés d'un chemin de propagation dans cet état. Ces conditions portent sur différents intervalles définies selon les caractéristiques de l'occurrence de la source, selon l'instant actuel et selon les propriétés du chemin. On prouve cette proposition.

Démonstration. On suppose que l'on a une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et un chemin P de la variable x vers la variable y bien défini par $\mathcal{A}rchi$. On suppose que l'on a une occurrence de la source x du chemin P apparue à l'instant upd et conservée pendant dur sur la source. Soit une exécution σ vérifiant la spécification et un état, σ_i, on suppose que l'on a :

$$\left(\begin{array}{l}\exists c \in \mathcal{C}(\mathbb{N}) : \\ \hat{\hat{x}}.\sigma_{c(i+1)} = upd \wedge \left(\begin{array}{c} Latency(P,c,\delta_{Lat},\Delta_{Lat}).\sigma_i \wedge Lag(P,c,\delta_{Lag},\Delta_{Lag}).\sigma_i \\ \wedge Shift(P,c,\delta_{Shi},\Delta_{Shi}).\sigma_i \wedge Freshness(P,c,\delta_{Fre},\Delta_{Fre}).\sigma_i \\ \wedge Fitness(P,c,\delta_{Fit},\Delta_{Fit}).\sigma_i \wedge Stability(P,c,\delta_{Sta},\Delta_{Sta}).\sigma_i \end{array}\right)\end{array}\right)$$

On utilise une horloge c de $\mathcal{C}(\mathbb{N})$ vérifiant la conjonction des prédicats. Si cette horloge est utilisée pour caractériser le chemin P et que $\hat{\hat{x}}.\sigma_{c(i+1)} = upd$ alors c'est l'occurrence de la source apparue à cet instant qui est utilisée pour définir l'occurrence de y dans l'état σ_{i+1}.

Pour prouver cette proposition, on analyse chacun des prédicats d'un chemin pour les reformuler sous une forme équivalente. On commence par examiner à part le régime initial.

1. Si jamais on a $upd = 0$, le chemin est alors dans son régime initial et d'après les propriétés du chemin on a $T.\sigma_{i+1} < \Delta_P$. On a bien l'équivalence dans ce cas là.

2. On a pour la latence :

$$
\begin{aligned}
& Latency(P, c, \delta_{Lat}, \Delta_{Lat}).\sigma_i \\
\Leftrightarrow \quad & \text{\{par définition\}} \\
& \delta_{Lat} \leq T.\sigma_{i+1} - \hat{\hat{x}}.\sigma_{c(i+1)} < \Delta_{Lat} \\
\Leftrightarrow \quad & \text{\{réécriture\}} \\
& \delta_{Lat} + \hat{\hat{x}}.\sigma_{c(i+1)} \leq T.\sigma_{i+1} < \Delta_{Lat} + \hat{\hat{x}}.\sigma_{c(i+1)}
\end{aligned}
$$

On en déduit donc une condition nécessaire sur le temps dans l'état σ_{i+1} pour que la latence soit respectée dans l'état suivant. On a de plus la réciprocité. Si dans un état $i+1$, l'instant $T.\sigma_{i+1}$ est dans l'intervalle $[\delta_{Lat} + upd, \Delta_{Lat} + upd[$ défini par les caractéristiques de l'occurrence source et la spécification, alors la latence est respectée si l'occurrence de la source apparue à l'instant upd est utilisée. Donc, si $T.\sigma_{i+1}$ est dans l'intervalle $PathInterval$, alors la latence est bien vérifiée.

3. Concernant le lag, on a :

$$
\begin{aligned}
& Lag(P, c, \delta_{Lag}, \Delta_{Lag}).\sigma_i \\
\Leftrightarrow \quad & \text{\{par définition\}} \\
& \delta_{Lag} \leq \hat{\hat{y}}.\sigma_{i+1} - \hat{\hat{x}}.\sigma_{c(i+1)} < \Delta_{Lag} \\
\Leftrightarrow \quad & \text{\{réécriture\}} \\
& \delta_{Lag} + \hat{\hat{x}}.\sigma_{c(i+1)} \leq \hat{\hat{y}}.\sigma_{i+1} < \Delta_{Lag} + \hat{\hat{x}}.\sigma_{c(i+1)}
\end{aligned}
$$

On a $\hat{\hat{y}}.\sigma_{i+1} = T.\sigma_{i+1}$ lorsque l'on est dans l'état dans lequel on vient de faire la mise à jour. On a donc une condition nécessaire et suffisante sur les instants pendant lesquels on peut mettre à jour y avec une occurrence de x pour que le prédicat de lag soit vérifié. Cette condition sur le temps $T.\sigma_{i+1}$ est donnée uniquement dans le cas où on a une mise à jour et complète l'intervalle auquel doit appartenir T dans ce cas.

4. On s'intéresse maintenant au prédicat de fraîcheur, de décalage et de pérennité. On donne une preuve intermédiaire, on prouve tout d'abord que la satisfaction de ces trois prédicats est équivalente à :

$$
\begin{aligned}
& T.\sigma_{i+1} \in [max\begin{pmatrix} upd + \delta_{Fre}, \\ upd + dur - \Delta_{Fit} \end{pmatrix} + \delta_{Shi}..min\begin{pmatrix} upd + \Delta_{Fre}, \\ upd + dur - \delta_{Fit} \end{pmatrix} + \Delta_{Shi} - 1[\wedge \\
& [max\begin{pmatrix} upd + \delta_{Fre}, \\ upd + dur - \Delta_{Fit} \end{pmatrix}..min\begin{pmatrix} upd + \Delta_{Fre}, \\ upd + dur - \delta_{Fit} \end{pmatrix}[\neq \varnothing
\end{aligned}
$$

Pour prouver cette équivalence, on prouve l'implication des deux définitions dans les deux sens. On montre ensuite que cette expression est équivalente à la conjonction d'une condition sur le temps T et d'une condition sur dur.

4. A. On montre tout d'abord que les prédicats impliquent cette expression. On considère tout d'abord la fraîcheur :

$$
\begin{aligned}
& Freshness(P, c, \delta_{Fre}, \Delta_{Fre}).\sigma_i \\
\Leftrightarrow \quad & \text{\{par définition\}} \\
& \delta_{Fre} \leq T.\sigma_{c(i+1)} - \hat{\hat{x}}.\sigma_{c(i+1)} < \Delta_{Fre} \\
\Leftrightarrow \quad & \text{\{réécriture\}} \\
& \delta_{Fre} + \hat{\hat{x}}.\sigma_{c(i+1)} \leq T.\sigma_{c(i+1)} < \Delta_{Fre} + \hat{\hat{x}}.\sigma_{c(i)}
\end{aligned}
$$

De même pour le prédicat de pérennité :

$$
\begin{array}{ll}
& \mathrm{Fitness}(P, c, \delta_{Fit}, \Delta_{Fit}).\sigma_i \\
\Leftrightarrow & \quad \{\text{par définition}\} \\
& \delta_{Fit} < \hat{\hat{x}}.\sigma_{c(i+1)} + d_{\bar{x}}.\sigma_{c(i+1)} - T.\sigma_{c(i+1)} \leq \Delta_{Fit} \\
\Leftrightarrow & \quad \{\text{multiplication par -1 et sachant que } \delta_{Fit} \geq 0\} \\
& -\Delta_{Fit} \leq T.\sigma_{c(i+1)} - (\hat{\hat{x}}.\sigma_{c(i+1)} + d_{\bar{x}}.\sigma_{c(i+1)}) < -\delta_{Fit} \\
\Leftrightarrow & \quad \{\text{réécriture}\} \\
& \hat{\hat{x}}.\sigma_{c(i+1)} + d_{\bar{x}}.\sigma_{c(i+1)} - \Delta_{Fit} \leq T.\sigma_{c(i+1)} < \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - \delta_{Fit}
\end{array}
$$

Par conjonction des prédicats de fraîcheur et de pérennité on a donc l'inégalité équivalente :

$$
max\begin{pmatrix} \hat{\hat{x}}.\sigma_{c(i+1)} + \delta_{Fre}, \\ \hat{\hat{x}}.\sigma_{c(i+1)} + d_{\bar{x}}.\sigma_{c(i+1)} - \Delta_{Fit} \end{pmatrix} \leq T.\sigma_{c(i+1)} < min\begin{pmatrix} \hat{\hat{x}}.\sigma_{c(i+1)} + \Delta_{Fre}, \\ \hat{\hat{x}}.\sigma_{c(i+1)} + d_{\bar{x}}.\sigma_{c(i+1)} - \delta_{Fit} \end{pmatrix}
$$

On en déduit que cet intervalle est bien non vide. En utilisant la propriété de décalage, on sait que l'on a :

$$
\begin{array}{ll}
& \delta_{Shi} \leq T.\sigma_{i+1} - T.\sigma_{c(i+1)} < \Delta_{Shi} \\
\Rightarrow & \quad \{\text{réécriture}\} \\
& \delta_{Shi} \leq T.\sigma_{i+1} - T.\sigma_{c(i+1)} \leq \Delta_{Shi} - 1
\end{array}
$$

En ajoutant les deux inégalités, on obtient par implication :

$$
max\begin{pmatrix} \hat{\hat{x}}.\sigma_{c(i)} + \delta_{Fre}, \\ \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - \Delta_{Fit} \end{pmatrix} + \delta_{Shi} \leq T.\sigma_i < min\begin{pmatrix} \hat{\hat{x}}.\sigma_{c(i)} + \Delta_{Fre}, \\ \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} - \delta_{Fit} \end{pmatrix} + \Delta_{Shi} - 1
$$

La vérification des prédicats de fraîcheur, pérennité et décalage implique donc pour $T.\sigma_{i+1}$ l'appartenance à cet intervalle.

4. B. On prouve la réciprocité. On suppose que l'on a une occurrence de la source apparue en upd et conservée pendant dur. On prouve que si l'on a :

$$
\begin{array}{l}
T.\sigma_{i+1} \in [max\begin{pmatrix} upd + \delta_{Fre}, \\ upd + dur - \Delta_{Fit} \end{pmatrix} + \delta_{Shi} .. min\begin{pmatrix} upd + \Delta_{Fre}, \\ upd + dur - \delta_{Fit} \end{pmatrix} + \Delta_{Shi} - 1[\wedge \\
[max\begin{pmatrix} upd + \delta_{Fre}, \\ upd + dur - \Delta_{Fit} \end{pmatrix} .. min\begin{pmatrix} upd + \Delta_{Fre}, \\ upd + dur - \delta_{Fit} \end{pmatrix}[\neq \varnothing
\end{array}
$$

alors les propriétés de fraîcheur, pérennité et décalage sont vérifiées.

On a :

$$
max\begin{pmatrix} upd + \delta_{Fre}, \\ upd + dur - \Delta_{Fit} \end{pmatrix} + \delta_{Shi} \leq T.\sigma_{i+1} < min\begin{pmatrix} upd + \Delta_{Fre}, \\ upd + dur - \delta_{Fit} \end{pmatrix} + \Delta_{Shi} - 1
$$

Alors, il existe nécessairement un instant t tel que $t = T.\sigma_{i+1} - \delta$ avec $\delta \in [\delta_{Shi} .. \Delta_{Shi}[$ et :

$$
max\begin{pmatrix} upd + \delta_{Fre}, \\ upd + dur - \Delta_{Fit} \end{pmatrix} \leq t < min\begin{pmatrix} upd + \Delta_{Fre}, \\ upd + dur - \delta_{Fit} \end{pmatrix}
$$

On donne la démonstration de l'existence de l'instant t dans un lemme suivant cette démonstration. Pour chaque instant, il existe un état σ_j de l'exécution tel que $T.\sigma_j = t$. On peut choisir j tel que $c(i) = j$ car $T.\sigma_j \in [upd, upd + dur[$ et donc $\hat{\hat{x}}.\sigma_j = upd$. D'après la définition des horloges du chemins, pointer vers cet état c'est choisir d'utiliser cette occurrence pour définir l'occurrence de l'image y. Les prédicats de fraîcheur et de pérennité sont alors vérifiées. De plus on a $T.\sigma_{i+1} - T.\sigma_{c(i+1)} = \delta$ et $\delta \in [\delta_{Shi} .. \Delta_{Shi}[$. Le décalage est donc aussi vérifié.

4.C. On prouve désormais que l'expression :

$$T.\sigma_{i+1} \in [max\begin{pmatrix} upd+\delta_{Fre}, \\ upd+dur-\Delta_{Fit} \end{pmatrix}+\delta_{Shi}..min\begin{pmatrix} upd+\Delta_{Fre}, \\ upd+dur-\delta_{Fit} \end{pmatrix}+\Delta_{Shi}-1[\wedge$$
$$[max\begin{pmatrix} upd+\delta_{Fre}, \\ upd+dur-\Delta_{Fit} \end{pmatrix}..min\begin{pmatrix} upd+\Delta_{Fre}, \\ upd+dur-\delta_{Fit} \end{pmatrix}[\neq \varnothing$$

est équivalente à l'expression utilisée dans la proposition 38 :

$$T.\sigma_{i+1} \in [upd+\delta_{Fre}+\delta_{Shi}..upd+\Delta_{Fre}+\Delta_{Shi}-1[$$
$$\vee$$
$$dur \in [max\begin{pmatrix} \delta_{Fre}+\delta_{Fit}+1, \\ T.\sigma_{i+1}-upd+\delta_{Fit}-\Delta_{Shi}+2 \end{pmatrix}..min\begin{pmatrix} \Delta_{Fre}+\Delta_{Fit}-1, \\ T.\sigma_{i+1}-upd+\Delta_{Fit}-\delta_{Shi} \end{pmatrix}]$$

On a bien dans les deux expressions : $T..\sigma_{i+1} \in [upd+\delta_{Fre}+\delta_{Shi}, upd+\Delta_{Fre}+\Delta_{Shi}+1]$. On s'intéresse désormais aux conditions sur dur. On a :

$$T.\sigma_{i+1} \in [upd+dur-\Delta_{Fit}+\delta_{Shi}..upd+dur-\delta_{Fit}+\Delta_{Shi}-1[\wedge$$
$$[max\begin{pmatrix} upd+\delta_{Fre}, \\ upd+dur-\Delta_{Fit} \end{pmatrix}..min\begin{pmatrix} upd+\Delta_{Fre}, \\ upd+dur-\delta_{Fit} \end{pmatrix}[\neq \varnothing$$

Pour que l'intervalle ne soit pas vide, on a nécessairement $upd+\delta_{Fre} < upd+\Delta_{Fre}$ et $upd+dur-\Delta_{Fit} < upd+dur-\delta_{Fit}$ car $\delta_{Fre} < \Delta_{Fre}$ et $\delta_{Fit} < \Delta_{Fit}$. On en déduit que pour que l'intervalle ne soit pas vide on a nécessairement :

$$\begin{array}{ll} & upd+\delta_{Fre} < upd+dur-\delta_{Fit} \wedge upd+dur-\Delta_{Fit} < upd+\Delta_{Fre} \\ \Leftrightarrow & \{\text{réécriture}\} \\ & \delta_{Fre}+\delta_{Fit} < dur \wedge dur < \Delta_{Fre}+\Delta_{Fit} \\ \Leftrightarrow & \{\text{réécriture}\} \\ & \delta_{Fre}+\delta_{Fit}+1 \leq dur \wedge dur \leq \Delta_{Fre}+\Delta_{Fit}-1 \end{array}$$

On a l'équivalence, donc si dur vérifie cette inégalité, l'intervalle est non vide. On suppose désormais que l'on a :

$$\begin{array}{ll} & T.\sigma_{i+1} \in [upd+dur-\Delta_{Fit}+\delta_{Shi}, upd+dur-\delta_{Fit}+\Delta_{Shi}-1[\\ \Leftrightarrow & \{\text{réécriture}\} \\ & upd+dur-\Delta_{Fit}+\delta_{Shi} \leq T.\sigma_{i+1} < upd+dur-\delta_{Fit}+\Delta_{Shi}-1 \\ \Leftrightarrow & \{\text{réécriture}\} \\ & dur \leq T.\sigma_{i+1}-upd+\Delta_{Fit}-\delta_{Shi} \wedge T.\sigma_{i+1}-upd+\delta_{Fit}-\Delta_{Shi}+1 < dur \\ \Leftrightarrow & \{\text{réécriture}\} \\ & T.\sigma_{i+1}-upd+\delta_{Fit}-\Delta_{Shi}+1 < dur \leq T.\sigma_{i+1}-upd+\Delta_{Fit}-\delta_{Shi} \\ \Leftrightarrow & \{\text{réécriture}\} \\ & T.\sigma_{i+1}-upd+\delta_{Fit}-\Delta_{Shi}+2 \leq dur \leq T.\sigma_{i+1}-upd+\Delta_{Fit}-\delta_{Shi} \end{array}$$

On a donc bien l'équivalence.

5. Le prédicat de stabilité, ne définit pas d'intervalle pour les valeurs de $T.\sigma_{c(i+1)}$. On ajoute la condition sur $d_{\bar{x}}.\sigma_{c(i+1)}$ issue de la définition de la stabilité dans la définition de l'intervalle $DurationInterval$. □

On démontre le lemme utilisé dans la démonstration précédente :

Lemme . *Soit deux entiers* a, b *tel que l'intervalle* $[a..b[$ *est non vide et* a *strictement positif. Soit deux entiers* δ, Δ *tel que* $\delta < \Delta$. *Alors on a :*

$$t \in [a+\delta..b+\Delta-1[\Rightarrow \exists t' \in \mathbb{N}, \delta' \in [\delta..\Delta[: t' = t-\delta' \wedge t' \in [a..b[$$

Démonstration. Soit deux entiers a, b tel que l'intervalle $[a..b[$ est non vide et a strictement positif et deux entiers δ, Δ tel que $\delta < \Delta$. On utilise pour construire t', une fonction $\delta'(t)$ définit à partir des valeurs de t.

$$\delta'(t) = t - a - \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor$$

On analyse d'abord les valeurs de δ' avant de définir un t' vérifiant le lemme en utilisant de δ'.

1. On veut prouver que pour $t \in [a+\delta..b+\Delta-1[$, la valeur de δ' est dans l'intervalle $[\delta, \Delta[$. Pour cela, on donne un encadrement de cette fonction. Par définition de la partie entière, on a :

$$\begin{array}{l} \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) - 1 < \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor \leq \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \\ \Rightarrow \quad \{\text{ajout de } a\} \\ a + \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) - 1 < a + \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor \leq a + \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \\ \Rightarrow \quad \{\text{multiplication par } -1\} \\ -a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \leq -a - \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < -a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) + 1 \\ \Rightarrow \quad \{\text{ajout de } t\} \\ t - a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \leq t - a - \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < t - a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) + 1 \end{array}$$

On démontre que la fonction $t \to (t - a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a))$ est croissante sur $\mathbb{R} \cap [a+\delta..b+\Delta]$ et donc sur $\mathbb{N} \cap [a+\delta..b+\Delta]$. En effet, sa dérivée est égal à :

$$1 - \frac{b-a}{b+\Delta-1-a-\delta}$$

Cette dérivée est positive car on a :

$$\begin{array}{l} \Delta - \delta \geq 1 \\ \Rightarrow \quad \{\text{ajout de } b-a > 0\} \\ b + \Delta - 1 - a - \delta \geq b - a \end{array}$$

et donc la fonction est strictement croissante. De cette croissance, on en déduit que les bornes de l'inégalité suivante sont deux fonctions croissantes :

$$t - a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \leq t - a - \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < t - a - \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) + 1$$

La longueur de l'intervalle est égale à 1 et c'est un intervalle ouvert à droite. Dans le domaine des entiers, il n'y a donc qu'un choix à chaque fois. On en déduit que la fonction qui définit δ' à partir de t est une fonction croissante sur $\mathbb{N}$. De plus pour les valeur de t dans $[a+\delta..b+\Delta-1[$, δ' est donc borné par :

$$\begin{array}{l} a + \delta - a - \frac{a+\delta-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \leq t - a - \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < b + \Delta - 1 - a - \frac{b+\Delta-1-a-\delta}{b+\Delta-1-a-\delta} * (b-a) + 1 \\ \Rightarrow \quad \{\text{simplification}\} \\ \delta \leq t - a - \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < \Delta \end{array}$$

Les valeurs de δ' sont donc dans l'intervalle $[\delta..\Delta[$.

2. On définit t' par :

$$t' = t - \delta'(t) = a + \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor$$

On prouve que cette valeur est comprise entre a et b. On a :

$$
\begin{array}{ll}
 & a + \delta \leq t < b + \Delta - 1 \\
\Leftrightarrow & \quad \{\text{réécriture}\} \\
 & 0 \leq t - a - \delta < b + \Delta - 1 - a - \delta \\
\Leftrightarrow & \quad \{\text{division par } b + \Delta - a - \delta \text{ strictement positif}\} \\
 & 0 \leq \frac{t-a-\delta}{b+\Delta-1-a-\delta} < 1 \\
\Leftrightarrow & \quad \{\text{multiplication par } b - a \text{ non nul}\} \\
 & 0 \leq \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) < b - a \\
\Leftrightarrow & \quad \{\text{propriété de la partie entière et } b - a \text{ entier}\} \\
 & 0 \leq \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < b - a \\
\Leftrightarrow & \quad \{\text{réécriture}\} \\
 & a \leq a + \lfloor \frac{t-a-\delta}{b+\Delta-1-a-\delta} * (b-a) \rfloor < b
\end{array}
$$

Donc on a bien t' compris dans l'intervalle $[a..b[$. □

A noter que la définition des propriétés pourrait être simplifiée. En effet, la vérification du prédicat de stabilité peut se faire uniquement lors d'une transition impliquant une mise à jour. Si ces propriétés sont vérifiées lors de cette transition alors elles sont vérifiées dans tous les états où l'on conserve cette occurrence de la source.

Pour construire une exécution vérifiant les propriétés d'un chemin, il s'agit donc en chaque état de choisir une occurrence vérifiant les propriétés données dans la proposition 38, tout en s'assurant de choisir ces occurrences dans l'ordre chronologique.

CONDITIONS D'ANTICIPATION SUR LE FUTUR La définition d'une relation de transition permet de construire les exécutions en suivant l'ordre chronologique, états par états. Pour une variable x, la variable associée $d_{\bar{x}}$ donne dans chaque état la durée pendant laquelle l'occurrence courante est conservée. Lors de la construction d'une exécution, on ne peut donc pas fixer cette valeur tant qu'on ne pas les états postérieurs. Si lors d'une transition, la variable x est mise à jour, alors fixer la valeur de la variable stabilité $d_{\bar{x}}$, c'est anticiper la date de la prochaine mise à jour. Cependant, la possibilité de mettre à jour dépend de l'évolution des autres variables du système et notamment de la disponibilité d'occurrences valides des sources. En anticipant la valeur de la variable $d_{\bar{x}}$ au début d'une occurrence, on peut anticiper une date de mise à jour qui est incompatible avec l'évolution future des autres variables.

Pour cette raison, on considère que lors de la définition de la relation de transition d'une variable, on ne connaît pas la valeur de la variable stabilité et la durée de l'occurrence de cette variable. La seule information dont l'on dispose pour une variable x est que l'on a :

$$\forall\, i \in \mathbb{N} : d_{\bar{x}}.\sigma_i \geq T.\sigma_i + 1 - \hat{\bar{x}}.\sigma_i$$

Autrement dit, tant que l'on a pas mis à jour, la valeur de la variable stabilité augmente. Pour les propriétés de sporadicité et de périodicité, on a vu que l'on pouvait donner les conditions pour que ces propriétés soient respectées sans connaître la valeur de $d_{\bar{x}}$. En décidant de mettre à jour ou pas, il s'agit justement de construire la valeur de cette variable dans les états antérieurs. Ne pas mettre à jour revient à faire croître cette variable et mettre à jour revient à stopper cette croissance et fixer définitivement la valeur de la stabilité dans les états antérieurs.

Dans le cas des propriétés d'un chemin, la satisfaction des prédicats de stabilité et de pérennité dépend de la durée des occurrences de la source. Si le décalage introduit le

long du chemin est suffisamment élevé, au moment où une occurrence est utilisée, il y a eu une mise à jour de la source et ce n'est plus cette occurrence qui est en cours. Sa durée est donc connue. Dans ce cas, la vérification des propriétés du chemin est possible. Si le décalage n'est pas suffisamment élevé, l'occurrence utilisée pour mettre à jour l'image est l'occurrence actuelle de la source. On ne connaît alors pas encore la durée de cette occurrence et on ne peut pas vérifier la validité de la mise à jour de l'image avec cette occurrence de la source. On anticipe alors cette validité.

Afin de valider cette anticipation, on déduit de l'utilisation d'une occurrence en cours de la source une condition d'anticipation sur la durée de cette occurrence et la variable stabilité. Une telle condition, comme les propriétés de sporadicité où de périodicité, définit un intervalle et donc les instants auxquels où peut ou où on doit mettre à jour cette source. C'est pour mettre en évidence cette condition sur la durée d'une occurrence que les conditions pour la satisfaction des propriétés d'un chemin de la proposition 38 sont définies par la conjonction d'une condition sur l'instant courant et d'une condition sur la durée de l'occurrence actuelle.

6.3 VARIABLES AUXILIAIRES

On introduit des variables servant à stocker les valeurs nécessaires à l'expression des conditions de mise à jour. On introduit de plus une variable permettant de donner dans certains cas des conditions d'anticipation sur l'évolution d'une variable si une de ces occurrences est utilisée pour mettre à jour l'image d'un chemin tout en étant toujours en cours. Ces variables sont utilisées pour définir l'action de chaque variable du système.

6.3.1 *Historique des occurrences*

D'après les reformulations des propriétés d'une spécification, afin de déterminer si l'on peut ou non mettre à jour une variable, la première information utile est la valeur de T dans l'état courant. On doit ensuite connaître la date de la dernière mise à jour de l'image. Finalement, on doit connaître les caractéristiques temporelles des occurrences utilisées pour définir l'occurrence actuelle de l'image. Ces caractéristiques sont, pour chaque occurrence liée ou pouvant être liée à l'image, la date d'apparition de cette occurrence et la durée pendant laquelle elle a été conservée.

On stocke ces informations dans des variables auxiliaires permettant à partir de l'état courant de consulter les caractéristiques temporelles des occurrences passées. On définit tout d'abord ce que sont les caractéristiques d'une occurrence.

Définition 63. *Caractéristiques d'une occurrence Les caractéristiques d'une occurrence sont un triplet appartenant à l'ensemble* $\mathbb{N} \times \mathbb{N} \times \mathcal{P}\mathtt{ath}$ *où* $\mathcal{P}\mathtt{ath}$ *est l'ensemble des chemins.*

Les deux premiers éléments d'un tel triplet ont pour but d'enregistrer la date d'apparition et la durée de l'occurrence. Le troisième élément du triplet est un chemin car on s'intéresse aux caractéristiques des occurrences dans le cadre des propriétés temporelles des chemins. On donne comme nom générique à un tel triplet o.

Définition 64. *Caractéristique de l'occurrence de la source d'un chemin. Pour une exécution* σ*, un triplet* o *donne les caractéristiques connues dans un état* σ_j *de la source d'un chemin* P *dans l'état* σ_i *tel que* $i \leq j$ *si le prédicat* $\mathtt{Charac}(o, i, P)$ *est vérifié avec :*

$$\begin{aligned}\mathtt{Charac}(o,i,P).\sigma_j \triangleq\ & \exists x, \exists P' : P = [x] : P' \ \wedge\ \forall\, c \in \mathcal{C}(P).\sigma : \\ & (\hat{\hat{x}}.\sigma_{c(i)} \neq \hat{\hat{x}}.\sigma_j \ \wedge\ o = \langle \hat{\hat{x}}.\sigma_{c(i)}, d_{\bar{x}}.\sigma_{c(i)}, P\rangle) \vee \\ & (\hat{\hat{x}}.\sigma_{c(i)} = \hat{\hat{x}}.\sigma_j \ \wedge\ o = \langle \hat{\hat{x}}.\sigma_{c(i)}, T.\sigma_j + 1 - \hat{\hat{x}}.\sigma_i, P\rangle)\end{aligned}$$

Un triplet vérifiant ce prédicat caractérise la source d'un chemin dans un état σ_i. Il contient donc les caractéristiques de la source du chemin utilisé pour construire l'occurrence de l'image du chemin dans cet état σ_i. On se place dans le contexte d'un état σ_j. Dans le cas où l'occurrence de la source qui est utilisée en σ_i est l'occurrence de la source dans l'état σ_j, on ne connaît pas sa durée. Dans ce cas la différence $T.\sigma_j - \hat{\hat{x}}.\sigma_i$ qui est une borne inférieure à cette durée est utilisée.

Une variable peut être image de plusieurs chemins, on s'intéresse aux caractéristiques de l'ensemble des sources des chemins dont cette variable est l'image.

Définition 65. *Ensemble des occurrences définissant une variable. Pour une exécution* σ *et un ensemble de chemins* $\mathcal{P}\mathtt{aths}$*. Pour une variable* x*, on définit dans un état* σ_j *l'ensemble des occurrences liées à* x *comme sources de chemins de* $\mathcal{P}\mathtt{aths}$ *et dans l'état* σ_i *:*

$$\mathtt{SrcCharac}(x, i, \mathcal{P}\mathtt{aths}).\sigma_j = \{o \mid \exists P \in \mathcal{P}\mathtt{aths} : P = P' : [x] \ \wedge\ \mathtt{Charac}(o, i, P).\sigma_j\}$$

Un tel ensemble contient toutes les caractéristiques des occurrences connues dans l'état σ_j et qui sont liées dans un état σ_i à la variable x par un chemin de $\mathcal{P}\mathtt{aths}$. Dans la pratique, cet ensemble de chemin $\mathcal{P}\mathtt{aths}$ est bien sûr l'ensemble des chemins définissant les propriétés de la spécification.

Une variable, si elle est image d'un chemin, est avant tout image d'une observation. Lorsque l'on veut mettre à jour l'image d'une observation, on choisit quelles occurrences des sources utiliser. L'image est alors liée aux occurrences d'autres variables à travers les occurrences des sources choisies. Étant donné que l'observation respecte la chronologie, les occurrences des sources qui servent à mettre à jour l'image sont toutes celles apparues après celles actuellement utilisées.

On définit une variable historique qui contient des ensembles de caractéristiques d'occurrences. Chaque ensemble contient toutes les caractéristiques des occurrences ayant servi à construire les sources dans un état. La variable historique contient les ensemble d'occurrences caractérisant chaque état situé entre l'état pointé par l'horloge de l'observation et l'état actuel.

Définition 66. *Historique des occurrences. Soit une spécification* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop}\rangle$*, et une exécution* σ *satisfaisant cette spécification. Pour une variable* y*, image d'une observation* $y \trianglelefteq f(X)$*, on définit la variable historique comme un tableau d'ensembles de caractéristiques d'occurrence tel que pour tout entier* i *:*

$$k = \max\{c(i) | c \in \mathcal{C}(y \trianglelefteq f(X)).\sigma\}$$

et

$$\begin{aligned}&\forall j \in [k..i] : H_{y \trianglelefteq f(X)}[j].\sigma_i \triangleq \\ &\qquad \left\{o \,\middle|\, \begin{array}{l} x \in X \ \wedge\ \exists o' \in \mathtt{SrcCharac}(x, j, \mathcal{P}\mathtt{aths}).\sigma_i : \\ o' = \langle \mathtt{upd}, \mathtt{dur}, P\rangle \ \wedge\ o = \langle \mathtt{upd}, \mathtt{dur}, P : [y]\rangle \end{array} \right\} \cup \\ &\left\{o \,\middle|\, x \in X \ \wedge\ \left(\begin{array}{l} (\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_j \ \wedge\ o = \langle \hat{\hat{x}}.\sigma_j, d_{\bar{x}}.\sigma_j, [x, y]\rangle) \vee \\ (\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_j \ \wedge\ o = \langle \hat{\hat{x}}.\sigma_j, T.\sigma_i + 1 - \hat{\hat{x}}.\sigma_j, [x, y]\rangle) \end{array} \right)\right\}\end{aligned}$$

avec :

$$\mathcal{P}aths = \{P : [x] \mid x \in X \ \wedge \ (P : [x, y]) \in \mathcal{P}aths(\mathcal{A}rchi)\}$$

Chaque élément du tableau $H_{y \preccurlyeq f(X)}$ contient les caractéristiques des occurrences des variables de X dans un état ainsi les caractéristiques des occurrences liées aux occurrences des variables de X dans ce même état. Les états concernés sont ceux entre $\sigma_{c(i)}$ et σ_i. De par la définition de la relation d'observation et la croissance de l'horloge de l'observation, ce sont les occurrences des sources de l'observation dans ces états qui peuvent être utilisées pour mettre à jour y. Si pour un de ces ensembles, les caractéristiques des différentes occurrences vérifient les propriétés des chemins alors les occurrences correspondantes des sources peuvent être utilisées pour mettre à jour l'image.

Concernant les indices du tableau, on commence par le maximum des indices pointés par une horloge de l'observation. Si l'on choisit une autre horloge de l'observation et comme toutes ces horloges pointent vers les mêmes occurrences des sources, les premiers éléments du tableau seraient des doublons. Les rangs suivants du tableau peuvent pour les mêmes raisons eux aussi contenir des doublons.

Ces caractéristiques sont celles connues dans l'état σ_i. C'est pour cette raison que la définition 64 donne les caractéristiques des occurrences dans un état tel que connues dans un autre état.

Proposition . *On définit les fonctions* IndexMin *et* IndexMax *qui donne les indices minimum et maximum d'un tableau. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$*, et une exécution* σ *satisfaisant cette spécification. Pour une variable* y*, image d'une observation* $y \preccurlyeq f(X)$*, on a alors :*

$$\forall\, i \in \mathbb{N} : max\{c(i) | c \in \mathcal{C}(y \preccurlyeq f(X)).\sigma\} = IndexMin(H_{y \preccurlyeq f(X)}.\sigma_i)$$

Le premier élément de ce tableau contient les caractéristiques des occurrences auxquelles l'image y est actuellement liée. On a la propriété suivante :

Proposition 39. *Minimum de l'historique des occurrences. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et une exécution* σ *satisfaisant cette spécification. Pour une variable* y*, image d'une observation* $y \preccurlyeq f(X)$*, la variable historique des occurrences vérifie la propriété suivante :*

$$\forall\, i \in \mathbb{N} : min(H_{y \preccurlyeq f(X)}.\sigma_i) = SrcCharac(y, i, \mathcal{P}aths(\mathcal{A}rchi)).\sigma_i$$

où $min(H_{y \preccurlyeq f(X)}.\sigma_i) = H_{y \preccurlyeq f(X)}[IndexMin(H_{y \preccurlyeq f(X)}.\sigma_i)].\sigma_i$.

Le minimum de l'historique des occurrences est donc l'ensemble des caractéristiques des occurrences utilisées pour définir l'occurrence actuelle de y.

Démonstration. Par définition et pour un état σ_i, on a :

$$\begin{aligned} &min(H_{y \preccurlyeq f(X)}.\sigma_i) = \\ &\qquad \left\{ o \;\middle|\; \begin{array}{l} x \in X \ \wedge \ \exists o' \in SrcCharac(x, c(i), \mathcal{P}aths).\sigma_i : \\ o' = \langle upd, dur, P \rangle \ \wedge \ o = \langle upd, dur, P : [y] \rangle \end{array} \right\} \ \cup \\ &\left\{ o \;\middle|\; x \in X \ \wedge \ \left(\begin{array}{l} (\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_{c(i)} \ \wedge \ o = \langle \hat{\hat{x}}.\sigma_{c(i)}, d_{\bar{x}}.\sigma_{c(i)}, [x, y] \rangle) \vee \\ (\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{c(i)} \ \wedge \ o = \langle \hat{\hat{x}}.\sigma_{c(i)}, T.\sigma_i + 1 - \hat{\hat{x}}.\sigma_{c(i)}, [x, y] \rangle) \end{array} \right) \right\} \end{aligned}$$

avec

$$\mathcal{P}aths = \{P : [x] \mid x \in X \ \wedge \ (P : [x, y]) \in \mathcal{P}aths(\mathcal{A}rchi)\}$$

et c tel que $c(i) = max\{c'(i) | c' \in \mathcal{C}(y \lessdot f(X)).\sigma\}$.

Pour chaque x appartenant à X, on a :

$$SrcCharac(x, c(i), \mathcal{P}aths).\sigma_i = \{o \mid \exists P \in \mathcal{P}aths : P = P' : [x] \wedge Charac(o, c(i), P).\sigma_i\}$$

En utilisant la définition de $\mathcal{P}aths$ on a alors :

$$\begin{array}{l} SrcCharac(x, c(i), \mathcal{P}aths).\sigma_i = \\ \{o \mid \exists P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) : P : [y] = P' : [x, y] \wedge Charac(o, c(i), P).\sigma_i\} \end{array}$$

Pour un élément de cet ensemble et un chemin $P : [y]$ de $\mathcal{P}aths(\mathcal{A}rchi)$ tel que $P : [y] = P' : [x, y]$:

$$\begin{array}{ll} & Charac(o, c(i), P).\sigma_i \\ \Rightarrow & \quad \{\text{par définition}\} \\ & \exists P'' : P = [z] : P'' \wedge \forall c' \in \mathcal{C}(P).\sigma : \\ & (\hat{\hat{z}}.\sigma_{c'(c(i))} \neq \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c'(c(i))}, d_{\bar{z}}.\sigma_{c'(c(i))}, P\rangle) \vee \\ & (\hat{\hat{z}}.\sigma_{c'(c(i))} = \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c'(c(i))}, T.\sigma_i - \hat{\hat{z}}.\sigma_{c'(c(i))}, P\rangle) \end{array}$$

On a $c \in \mathcal{C}([x, y]).\sigma$, $c' \in \mathcal{C}(P).\sigma$ et $P : [y] = P' : [x, y]$ donc d'après la proposition 22 on a $c' \circ c \in \mathcal{C}(P' : [x, y]).\sigma$. On en déduit que l'on a :

$$\begin{array}{l} \exists P'' : P : [y] = [z] : P'' : [y] \wedge \exists c'' \in \mathcal{C}(P : [y]).\sigma : \\ (\hat{\hat{z}}.\sigma_{c''(i)} \neq \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c''(i)}, d_{\bar{z}}.\sigma_{c''(i)}, P\rangle) \vee \\ (\hat{\hat{z}}.\sigma_{c''(i)} = \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c''(i)}, T.\sigma_i - \hat{\hat{z}}.\sigma_{c''(i)}, P\rangle) \end{array}$$

Pour un tel élément, on a dans l'historique un triplet correspondant où seul le chemin est modifié. C'est donc un élément vérifiant :

$$\begin{array}{l} \exists P'' : P : [y] = [z] : P'' : [y] \wedge \exists c'' \in \mathcal{C}(P : [y]).\sigma : \\ (\hat{\hat{z}}.\sigma_{c''i)} \neq \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c''i)}, d_{\bar{z}}.\sigma_{c''(i)}, P : [y]\rangle) \vee \\ (\hat{\hat{z}}.\sigma_{c'(c(i))} = \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c''(i)}, T.\sigma_i - \hat{\hat{z}}.\sigma_{c''(i)}, P : [y]\rangle) \end{array}$$

Or d'après la proposition 20 pour toutes les horloges du chemin, on a la même valeur de profil temporel et la même durée dans l'état pointé par ces horloges. On a donc :

$$\begin{array}{ll} & \exists P'' : P : [y] = [z] : P'' : [y] \wedge \forall c'' \in \mathcal{C}(P : [y]).\sigma : \\ & (\hat{\hat{z}}.\sigma_{c''i)} \neq \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c''i)}, d_{\bar{z}}.\sigma_{c''(i)}, P : [y]\rangle) \vee \\ & (\hat{\hat{z}}.\sigma_{c'(c(i))} = \hat{\hat{z}}.\sigma_i \wedge o = \langle \hat{\hat{z}}.\sigma_{c''(i)}, T.\sigma_i - \hat{\hat{z}}.\sigma_{c''(i)}, P : [y]\rangle) \\ \Rightarrow & \quad \{\text{par définition}\} \\ & Charac(o, i, P : [y]).\sigma \end{array}$$

On en déduit que l'on a :

$$\left\{o \;\middle|\; \begin{array}{l} x \in X \wedge \exists o' \in SrcCharac(x, c(i), \mathcal{P}aths).\sigma_i : \\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \end{array}\right\} =$$
$$\left\{o \;\middle|\; \begin{array}{l} \exists x \in X : \exists P \in \mathcal{P}aths(\mathcal{A}rchi) : \; P : [y] = P' : [x, y] \\ \wedge Charac(o, i, P : [y]).\sigma_i \end{array}\right\}$$

Or dans la spécification, il n'existe qu'une seule observation ayant y comme image. Les seuls chemins de l'architecture ayant y comme image sont ceux passant par les sources X de cette observation. Dans l'ensemble précédent, on ne s'intéresse cependant qu'au chemin de longueur supérieure ou égale à 3. Les autres chemins ayant y comme image sont ceux d'une variable de X à y.

On considère la seconde partie du minimum de l'historique :

$$\left\{ o \middle| x \in X \wedge \left(\begin{array}{l} (\hat{\hat{x}}.\sigma_i \neq \hat{\hat{x}}.\sigma_{c(i)} \ \wedge \ o = \langle \hat{\hat{x}}.\sigma_{c(i)}, d_{\hat{x}}.\sigma_{c(i)}, [x,y] \rangle) \vee \\ (\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{c(i)} \ \wedge \ o = \langle \hat{\hat{x}}.\sigma_{c(i)}, T.\sigma_i + 1 - \hat{\hat{x}}.\sigma_{c(i)}, [x,y] \rangle) \end{array} \right) \right\}$$

Cet ensemble, par définition de $Charac(o, i, [x, y]))$ pour $x \in X$ est égal à :

$$\{o | x \in X \ \wedge \ Charac(o, i, [x, y])\}$$

L'union des deux ensemble donne donc bien l'ensemble suivant :

$$\{o \mid \exists P \in \mathcal{P}aths(\mathcal{A}rchi) : P = P' : [y] \ \wedge \ Charac(o, i, P).\sigma_i\} = SrcCharac(y, i, \mathcal{P}aths(\mathcal{A}rchi)).\sigma_i$$

□

On a de plus la proposition suivante qui indique que le minimum de la variable histoire est modifiée dans le cas où la variable est mise à jour.

Proposition 40. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ telle qu'il existe une observation $y \lesssim f(X)$ dans l'architecture du système. Soit σ une exécution satisfaisant cette spécification. La relation de transition de la variable y vérifie la propriété suivante :*

$$\forall i \in \mathbb{N} : (\hat{\hat{y}}.\sigma_i \neq \hat{\hat{y}}.\sigma_{i+1} \Leftrightarrow min(H_{y \lesssim f(X)}.\sigma_i) < min(H_{y \lesssim f(X)}.\sigma_{i+1}))$$

Démonstration. Cette proposition est déduite de la proposition 15 page 57 qui dit que si jamais l'image est mise à jour c'est avec de nouvelles occurrences des images. Le minimum de la variable histoire contenant les caractéristiques des occurrences couramment utilisées pour définir l'occurrence de l'image, si l'image est mise à jour, alors ces occurrences et leur caractéristiques changent et donc le minimum de la variable histoire est aussi modifié. □

On définit comment on construit la variable historique état par état. On a la proposition suivante :

Proposition 41. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et une exécution σ satisfaisant cette spécification. Pour une variable y, image d'une observation $y \lesssim f(X)$, la variable historique des occurrences liées aux sources X par les chemins de $\mathcal{P}aths(\mathcal{A}rchi)$ vérifie pour tout entier i :*

$$\forall j \in [IndexMin(H_{y \lesssim f(X)}.\sigma_i..i-1] : H_{y \lesssim f(X)}[j].\sigma_i = \left\{ o \middle| \begin{array}{l} \exists o' \in H_{y \lesssim f(X)}[j].\sigma_{i-1} : o' = \langle upd, dur, [z] : P \rangle \wedge \\ \left(\begin{array}{l} (\hat{\hat{z}}.\sigma_{i-1} \neq upd \ \wedge \ o = \langle upd, dur, [z] : P \rangle) \vee \\ (\hat{\hat{z}}.\sigma_{i-1} = upd \ \wedge \ o = \langle upd, T.\sigma_i + 1 - \hat{\hat{z}}.\sigma_{i-1}, [z] : P \rangle) \end{array} \right) \end{array} \right\}$$

et :

$$H_{y \lesssim f(X)}[i].\sigma_i = \{\langle \hat{\hat{x}}.\sigma_i, T.\sigma_i + 1 - \hat{\hat{x}}.\sigma_i, [x,y] \rangle | x \in X\} \cup \left\{ o \middle| \begin{array}{l} x \in X \ \wedge \ \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lesssim g(Z)}.\sigma_i) \\ \wedge \ o' = \langle upd, dur, P \rangle \ \wedge \ o = \langle upd, dur, P : [y] \rangle \ \wedge \ P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\}$$

Cette construction se fait donc en deux étapes. Premièrement on met à jour la durée des occurrences dans les caractéristiques présentes dans l'historique à l'état précédent selon que les caractéristiques correspondent à une occurrence en cours ou pas. Ensuite, on ajoute un nouvel indice au tableau correspondant à l'état courant. On insère à cet indice les caractéristiques des sources de l'observation dans l'état courant. Il s'agit donc, lors de chaque transition, de compléter la variable historique avec les caractéristiques de l'état courant.

Démonstration. Pour prouver cette proposition, on considère une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et une exécution σ satisfaisant cette spécification. On considère de plus une observation $y \lesssim f(X)$. On effectue une preuve par récurrence sur $\mathbb{N}$ pour prouver que la propriété est vérifiée en tout état de l'exécution.

1. Pour 0, on a, pour tout horloge c, $c(0) = 0$. Donc le tableau définit dans la proposition est réduit à un seul élément défini par :

$$\begin{array}{l} \{\langle \hat{\hat{x}}.\sigma_0, T.\sigma_0 + 1 - \hat{\hat{x}}.\sigma_0, [x]\rangle | x \in X\} \cup \\ \left\{ o \left| \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lesssim g(Z)}.\sigma_0) \\ \wedge\, o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right. \right\} \end{array}$$

On prouve que cet ensemble définit bien $H_{y \lesssim f(X)}.\sigma_0$ qui est aussi réduit à un seul élément : $H_{y \lesssim f(X)}[0]$ dans l'état initial. Pour cela, on s'intéresse à deux types de caractéristiques d'occurrences de variables liées à y.

1.A. On s'intéresse tout d'abord aux caractéristiques des occurrences sources de chemins de taille 2. Ces chemins sont ceux entre les variables de X et y. On s'intéresse donc aux caractéristiques des occurrences des variables de X. On considère l'ensemble suivant :

$$\{\langle \hat{\hat{x}}.\sigma_0, T.\sigma_0 + 1 - \hat{\hat{x}}.\sigma_0, [x]\rangle | x \in X\}$$

Dans la définition de $H_{y \lesssim f(X)}[0]$, le deuxième élément de la définition donne les caractéristiques des occurrences de X, soit :

$$\left\{ o \left| x \in X \wedge \left(\begin{array}{l} (\hat{\hat{x}}.\sigma_0 \neq \hat{\hat{x}}.\sigma_0 \wedge o = \langle \hat{\hat{x}}.\sigma_0, d_{\hat{x}}.\sigma_0, [x, y]\rangle) \vee \\ (\hat{\hat{x}}.\sigma_0 = \hat{\hat{x}}.\sigma_0 \wedge o = \langle \hat{\hat{x}}.\sigma_0, T.\sigma_0 + 1 - \hat{\hat{x}}.\sigma_0, [x, y]\rangle) \end{array} \right) \right. \right\}$$

Étant donné que dans l'état initial pour tout chemin $[x, y]$ et pour toute horloge c d'un tel chemin on a $c(0) = 0$, on a donc $\hat{\hat{x}}.\sigma_0 = \hat{\hat{x}}.\sigma_{c(0)}$. Il y a bien égalité des deux ensembles.

1.B. On s'intéresse désormais aux caractéristiques des occurrences des sources d'un chemin de taille supérieur ou égal à 3. Cet ensemble de caractéristiques est donné dans la proposition par :

$$\begin{array}{l} \left\{ o \left| \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lesssim g(Z)}.\sigma_0) \\ \wedge\, o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right. \right\} \end{array}$$

On prouve que cet ensemble est égal à :

$$\left\{ o \left| \begin{array}{l} x \in X \wedge \exists o' \in SrcCharac(x, 0, \mathcal{P}aths).\sigma_0 : \\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \end{array} \right. \right\}$$

avec :

$$\mathcal{P}aths = \{P : [x] \mid x \in X \wedge (P : [x, y]) \in \mathcal{P}aths(\mathcal{A}rchi)\}$$

On a :

$$
\begin{array}{ll}
 & \left\{ o \;\middle|\; \begin{array}{l} x \in X \ \wedge\ \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lesssim g(Z)}.\sigma_0) \\ \wedge\ o' = \langle upd, dur, P\rangle \ \wedge\ o = \langle upd, dur, P : [y]\rangle \ \wedge\ P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
= & \quad \{\text{proposition 39}\} \\
 & \left\{ o \;\middle|\; \begin{array}{l} x \in X \ \wedge\ \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in SrcCharac(x, 0, \mathcal{P}aths(\mathcal{A}rchi)).\sigma_0 \\ \wedge\ o' = \langle upd, dur, P\rangle \ \wedge\ o = \langle upd, dur, P : [y]\rangle \ \wedge\ P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
= & \quad \{\text{simplification}\} \\
 & \left\{ o \;\middle|\; \begin{array}{l} x \in X \ \wedge\ \exists o' \in SrcCharac(x, 0, \mathcal{P}aths(\mathcal{A}rchi)).\sigma_0 \\ \wedge\ o' = \langle upd, dur, P\rangle \ \wedge\ o = \langle upd, dur, P : [y]\rangle \ \wedge\ P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
= & \quad \{\text{définition de } \mathcal{P}aths\} \\
 & \left\{ o \;\middle|\; \begin{array}{l} x \in X \ \wedge\ \exists o' \in SrcCharac(x, 0, \mathcal{P}aths).\sigma_0 : \\ o' = \langle upd, dur, P\rangle \ \wedge\ o = \langle upd, dur, P : [y]\rangle \end{array} \right\}
\end{array}
$$

Tous les chemins menant à y passent par les variables de X. On a les caractéristiques de toutes les occurrences qui définissent les occurrences de X dans l'état 0. Ce sont ces occurrences qui sont liées à y à travers X par des chemins de taille supérieure ou égale à 3. On met de plus à jour le chemin dans les caractéristiques. L'ensemble vu avant complète la définition de $H_{y \lesssim f(X)}[0].\sigma_0$ avec les caractéristiques des occurrences de X.

2. On suppose que la proposition est vérifiée pour tout entier strictement inférieur à un entier i. On prouve alors que la proposition est vérifiée pour l'état i. On s'intéresse tout d'abord aux éléments du tableau avant le i – ème élément puis on s'intéresse à ce dernier élément.

2.A. On considère le j-ème élément du tableau tel que $j < i$. Les caractéristiques des occurrences dans cet élément sont stockées dans $H_{y \lesssim f(X)}[j].\sigma_{i-1}$, à l'exception des caractéristiques des occurrences qui étaient en cours et dont la durée doit être mise à jour. On prouve que la proposition propose bien de modifier les caractéristiques de ces occurrences et uniquement celles ci.

La proposition définit que pour chaque occurrence $\langle upd, dur, [z] : P\rangle$ de $H_{y \lesssim f(X)}[j].\sigma_{i-1}$, on stocke dans $H_{y \lesssim f(X)}[j].\sigma_{i-1}$, l'occurrence o tel que :

$$
\left(\begin{array}{l} (\hat{z}.\sigma_{i-1} \neq upd \ \wedge\ o = \langle upd, dur, [z] : P\rangle) \vee \\ (\hat{z}.\sigma_{i-1} = upd \ \wedge\ o = \langle upd, T.\sigma_i + 1 - \hat{z}.\sigma_{i-1}, [z] : P\rangle) \end{array} \right)
$$

Il y a quatre cas possibles selon qu'il y a une mise jour entre σ_{i-1} et σ_i :

- s'il y a une mise à jour :
  - si dans σ_{i-1}, on a $\hat{z}.\sigma_{i-1} \neq upd$, alors la mise à jour n'influence pas car ce n'était pas l'occurrence courante de z et la valeur dur de l'état antérieur est toujours correcte ;
  - si dans σ_{i-1}, on a $\hat{z}.\sigma_{i-1} = upd$, alors la mise à jour indique la fin de cette occurrence et on remplace dur avec sa valeur définitive ;
- s'il n'y a pas de mise à jour :
  - si dans σ_{i-1}, on a $\hat{z}.\sigma_{i-1} \neq upd$, on conserve la valeur de dur, l'occurrence étant fini dans un état antérieur σ_{i-1} et sa durée étant donc connue ;
  - si dans σ_{i-1}, on a $\hat{z}.\sigma_{i-1} = upd$, même si ce n'est pas la fin de cette occurrence, on effectue une mise à jour indiquant que l'occurrence a duré une unité de temps supplémentaire.

On réutilise donc les valeurs contenues dans $H_{y \lesssim f(X)}[j].\sigma_{i-1}$ en remettant à jour les occurrences dont la durée a augmenté.

2.B. Le i-ème élément du tableau correspond aux caractéristiques des occurrences des variables X et liées à ces variables dans l'état courant. On reprend la preuve donnée dans l'état initial.

On s'intéresse tout d'abord aux caractéristiques des occurrences sources de chemins de taille 2. Ces chemins sont ceux entre les variables de X et y. On s'intéresse donc aux caractéristiques des occurrences des variables de X. On considère l'ensemble suivant :

$$\{\langle \hat{\bar{x}}.\sigma_i, T.\sigma_i + 1 - \hat{\bar{x}}.\sigma_i, [x, y]\rangle | x \in X\}$$

Dans la définition de $H_{y \lesssim f(X)}[i].\sigma_i$, le deuxième élément de la définition donne les caractéristiques des occurrences de X, soit :

$$\begin{aligned}
&\left\{ o \middle| x \in X \wedge \left(\begin{array}{l} (\hat{\bar{x}}.\sigma_i \neq \hat{\bar{x}}.\sigma_i \ \wedge\ o = \langle \hat{\bar{x}}.\sigma_i, d_{\bar{x}}.\sigma_i, [x, y]\rangle) \vee \\ (\hat{\bar{x}}.\sigma_i = \hat{\bar{x}}.\sigma_i \ \wedge\ o = \langle \hat{\bar{x}}.\sigma_i, T.\sigma_i + 1 - \hat{\bar{x}}.\sigma_i, [x, y]\rangle) \end{array} \right) \right\} \\
= \quad & \{\text{réécriture}\} \\
& \{o | x \in X \wedge \hat{\bar{x}}.\sigma_i = \hat{\bar{x}}.\sigma_i \wedge o = \langle \hat{\bar{x}}.\sigma_i, T.\sigma_i + 1 - \hat{\bar{x}}.\sigma_i, [x, y]\rangle)\} \\
= \quad & \{\text{réécriture}\} \\
& \{\langle \hat{\bar{x}}.\sigma_i, T.\sigma_i + 1 - \hat{\bar{x}}.\sigma_i, [x, y]\rangle | x \in X\}
\end{aligned}$$

Il y a bien égalité des deux ensembles.

On s'intéresse désormais aux caractéristiques des occurrences des sources d'un chemin de taille supérieur ou égal à 3. Cet ensemble de caractéristiques est donné dans la proposition par :

$$\left\{ o \middle| \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lesssim g(Z)}.\sigma_i) \\ \wedge\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\}$$

On prouve que cet ensemble est égal à :

$$\left\{ o \middle| \begin{array}{l} x \in X \wedge \exists o' \in SrcCharac(x, i, \mathcal{P}aths').\sigma_i : \\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \end{array} \right\}$$

avec :

$$\mathcal{P}aths = \{P : [x] \mid x \in X \wedge (P : [x, y]) \in \mathcal{P}aths(\mathcal{A}rchi)\}$$

On a :

$$\begin{aligned}
& \left\{ o \middle| \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lesssim g(Z)}.\sigma_i) \\ \wedge\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [x]\rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
= \quad & \{\text{proposition 39}\} \\
& \left\{ o \middle| \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in SrcCharac(x, i, \mathcal{P}aths(\mathcal{A}rchi)).\sigma_i) \\ \wedge\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [x]\rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
= \quad & \{\text{simplification}\} \\
& \left\{ o \middle| \begin{array}{l} x \in X \wedge \exists o' \in SrcCharac(x, i, \mathcal{P}aths(\mathcal{A}rchi)).\sigma_i) \\ \wedge\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [x]\rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
= \quad & \{\text{définition de } \mathcal{P}aths\} \\
& \left\{ o \middle| \begin{array}{l} x \in X \wedge \exists o' \in SrcCharac(x, i, \mathcal{P}aths).\sigma_i : \\ o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \end{array} \right\}
\end{aligned}$$

Tous les chemins menant à y passent par les variables de X. On a les caractéristiques de toutes les occurrences qui définissent les occurrences de X dans l'état σ_i. Ce sont ces occurrences qui sont liées à y à travers X par des chemins de taille supérieure ou égale à 3. L'ensemble vu avant complète la définition de $H_{y \lesssim f(X)}[i].\sigma_i$ avec les caractéristiques des occurrences de X. □

6.3.2 *Conditions d'anticipation*

On définit pour une variable, source d'un chemin de propagation, une variable permettant de prendre en compte les conditions d'anticipation sur la durée des occurrences de cette variable. Une telle variable est un couple d'entier tel que ceux paramétrant la sporadicité d'une variable et bornant la durée d'une occurrence et donc la valeur de la variable stabilité.

Définition 67. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et une variable* x *source d'un ou plusieurs chemins. Alors pour une telle variable, on définit une variable condition d'anticipation* $\mathtt{dur}_x$ *sur la durée des occurrences de* x *comme un couple prenant ses valeurs dans* $\mathbb{N} \times \mathbb{N} \cup +\infty$ *le long d'une exécution.*

Cette variable sert à définir des conditions sur la durée des occurrences d'une variable en fonction des chemins dont elle est source et de l'utilisation des occurrentes courantes. Pour une variable, on définit de telles variables conditions pour chacun des chemins dont elle est la source. On a alors :

Définition 68. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et une variable* x *source d'un ou plusieurs chemins. Alors pour chaque chemin* P *dont* x *est la source ; on définit une variable condition d'anticipation* $\mathtt{dur}_P$ *sur la durée des occurrences de* x *comme un couple prenant ses valeurs dans* $\mathbb{N} \times \mathbb{N} \cup +\infty$ *le long d'une exécution.*

La valeur d'une variable condition d'anticipation d'une variable x de l'architecture est défini selon les variables condition d'anticipation des chemins dont x est l'image.

Définition 69. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et une variable* x *source d'un ou plusieurs chemins. Pour une exécution* σ*, on a alors :*

$$\forall i \in \mathbb{N} : \mathtt{dur}_x.\sigma_i \triangleq \left\langle \begin{array}{l} \min(\{\delta_{[x]:P} | \exists [x] : P \in \mathcal{P}aths(\mathcal{A}rchi) : \mathtt{dur}_{[x]:P}.\sigma_i = \langle \delta_{[x]:P}, \Delta_{[x]:P} \rangle\}), \\ \max(\{\Delta_{[x]:P} | \exists [x] : P \in \mathcal{P}aths(\mathcal{A}rchi) : \mathtt{dur}_{[x]:P}.\sigma_i = \langle \Delta_{[x]:P}, \Delta_{[x]:P} \rangle\}) \end{array} \right\rangle$$

On donne finalement dans chaque état la définition des variables de conditions d'anticipation pour chacun des chemins.

Définition 70. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et un chemin* P *d'une variable* x *source vers une variable* y*. On suppose que l'on a la propriété suivante sur le chemin* P*.*

$$P \left\{ \begin{array}{l} \mathtt{Latency}(\delta_{\mathtt{Lat}}, \Delta_{\mathtt{Lat}}), \mathtt{Lag}(\delta_{\mathtt{Lag}}, \Delta_{\mathtt{Lag}}), \mathtt{Shift}(\delta_{\mathtt{Shi}}, \Delta_{\mathtt{Shi}}), \\ \mathtt{Freshness}(\delta_{\mathtt{Fre}}, \Delta_{\mathtt{Fre}}), \mathtt{Fitness}(\delta_{\mathtt{Fit}}, \Delta_{\mathtt{Fit}}), \mathtt{Stability}(\delta_{\mathtt{Sta}}, \Delta_{\mathtt{Sta}}) \end{array} \right\}$$

Soit une exécution σ *satisfaisant la spécification, alors, dans tout état, la condition d'anticipation sur la durée des occurrences de la source est définie par :*

$$\forall i \in \mathbb{N} : \mathtt{dur}_P.\sigma_i \triangleq \left\{ \begin{array}{l} \langle \max\left(\begin{array}{c} \delta_{\mathtt{Sta}}, \delta_{\mathtt{Fre}} + \delta_{\mathtt{Fit}} + 1, \\ T.\sigma_i - \mathtt{upd} + \delta_{\mathtt{Fit}} - \Delta_{\mathtt{Shi}} + 2 \end{array} \right), \min\left(\begin{array}{c} \Delta_{\mathtt{Sta}} - 1, \Delta_{\mathtt{Fre}} + \Delta_{\mathtt{Fit}} - 1, \\ T.\sigma_i - \mathtt{upd} + \Delta_{\mathtt{Fit}} - \delta_{\mathtt{Shi}} \end{array} \right) \rangle \\ \textit{si } \hat{\hat{x}}.\sigma_{c(i)} = \hat{\hat{x}}.\sigma_i \wedge \forall c \in \mathcal{C}(P).\sigma : \hat{\hat{x}}.\sigma_{c(i-1)} \neq \hat{\hat{x}}.\sigma_i \\ \langle \max\left(\begin{array}{c} \delta, \delta_{\mathtt{Sta}}, \delta_{\mathtt{Fre}} + \delta_{\mathtt{Fit}} + 1, \\ T.\sigma_i - \mathtt{upd} + \delta_{\mathtt{Fit}} - \Delta_{\mathtt{Shi}} + 2 \end{array} \right), \min\left(\begin{array}{c} \delta, \Delta_{\mathtt{Sta}} - 1, \Delta_{\mathtt{Fre}} + \Delta_{\mathtt{Fit}} - 1, \\ T.\sigma_i - \mathtt{upd} + \Delta_{\mathtt{Fit}} - \delta_{\mathtt{Shi}} \end{array} \right) \rangle \\ \textit{si } \hat{\hat{x}}.\sigma_{c(i)} = \hat{\hat{x}}.\sigma_i \wedge \forall c \in \mathcal{C}(P).\sigma : \hat{\hat{x}}.\sigma_{c(i-1)} = \hat{\hat{x}}.\sigma_i \wedge \mathtt{dur}_P.\sigma_{i-1} = \langle \delta, \Delta \rangle \\ \langle 0, +\infty \rangle \textit{ sinon} \end{array} \right.$$

Justification. L'évolution de la condition d'un chemin dépend de l'occurrence de la source qui est utilisée pour définir l'occurrence de l'image du chemin. Dans le cas où c'est une occurrence terminée de la source qui est utilisée par l'image, il n'y a pas de condition d'anticipation. Si jamais c'est l'occurrence actuelle de la source qui est utilisée alors la durée de cette occurrence peut être contrainte par les propriété de pérennité, de fraîcheur et de stabilité. Cette contrainte est définie comme un intervalle d'appartenance de la variable stabilité défini par l'intervalle $\mathtt{DurationInterval}$ de la proposition 38. La contrainte dépend de chaque état où l'occurrence est utilisée et donc on s'intéresse à l'intersection de la condition d'anticipation donnée dans l'état courant et des conditions définies par les états antérieurs où l'on utilisait déjà cette occurrence. Lorsque l'image du chemin est liée à une nouvelle occurrence, la condition est reinitialisée.

6.4 SYSTÈME DE TRANSITIONS ÉQUIVALENT À UNE SPÉCIFICATION

En utilisant les propositions données précédemment et reformulant les propriétés d'une spécification et avec les variables auxiliaires, on définit un système de transitions équivalent à une spécification à base d'observations.

6.4.1 *Action associée à une variable*

On considère une spécification où l'architecture est bien définie, c'est-à-dire que l'ensemble des relations d'observations définies par la spécification est satisfiable. On s'intéresse au propriétés temporelles de la spécification et on considère ici que les possibilités de mise à jour des variables sont uniquement définies par ces propriétés. L'action d'une variable est donc uniquement définie par la conjonction des différentes propriétés temporelles. Le schéma de définition de l'action d'une variable diffère donc selon que la variable est sporadique, périodique ou ni l'un ni l'autre et aussi selon qu'elle est image d'une observation ou pas. On donne la définition dans chacun des cas. La définition de ces actions utilise les variables primées donnant la valeur d'une variable dans l'état suivant.

On commence par donner une action qui définit la condition d'anticipation d'une variable.

Définition 71. *Condition d'anticipation d'une variable. Soit une spécification* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop}\rangle$. *Pour une variable* x*, on définit l'action sur la condition d'anticipation de* x *par :*

$$A_{\mathtt{dur}_x} \triangleq \mathtt{dur}'_x = \left\langle \begin{array}{l} \mathtt{max}(\{\delta_{[x]:P} | \exists [x] : P \in \mathcal{P}\mathtt{aths}(\mathcal{A}\mathtt{rchi}) : \mathtt{dur}'_{[x]:P} = \langle \delta_{[x]:P}, \Delta_{[x]:P}\rangle\}), \\ \mathtt{min}(\{\Delta_{[x]:P} | \exists [x] : P \in \mathcal{P}\mathtt{aths}(\mathcal{A}\mathtt{rchi}) : \mathtt{dur}'_{[x]:P} = \langle \Delta_{[x]:P}, \Delta_{[x]:P}\rangle\}) \end{array} \right\rangle$$

On utilise cette action pour définir l'action d'une variable sporadique :

Définition 72. *Action d'une variable sporadique. Soit une spécification* $\langle \mathcal{A}\mathtt{rchi}, \mathcal{P}\mathtt{rop}\rangle$ *telle qu'il n'existe pas d'observation* $y \preccurlyeq f(X)$ *dans l'architecture du système. On suppose que la variable* y *est sporadique de paramètre* $\delta_{\mathtt{Spo}}, \Delta_{\mathtt{Spo}}$*. L'action de la variable* y *est définie par :*

$$\begin{array}{l} A_y \triangleq A_{\mathtt{dur}_y} \wedge \textit{Let } \langle \delta, \Delta\rangle = \mathtt{dur}'_y \textit{ in} \\ \left(\begin{array}{c} \hat{y}' \neq \hat{y} \wedge \mathtt{SporadicUpdate}(T, \hat{y}, \mathtt{min}(\delta, \delta_{\mathtt{Spo}}), \mathtt{max}(\Delta, \Delta_{\mathtt{Spo}})) \\ \vee \\ \hat{y}' = \hat{y} \wedge \mathtt{SporadicIdle}(T, \hat{y}, \mathtt{min}(\delta, \delta_{\mathtt{Spo}}), \mathtt{max}(\Delta, \Delta_{\mathtt{Spo}})) \end{array} \right) \end{array}$$

Ici, seule la sporadicité et les conditions d'anticipation guident le comportement de la variable.

Définition 73. *Action d'une variable périodique. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *tel qu'il n'existe pas d'observation* $y \preccurlyeq f(X)$ *dans l'architecture du système et tel qu'il existe une propriété de périodicité sur cette variable :*

$$y\ \{\mathtt{Periodic}(P, J, \phi)\} \in \mathcal{P}rop$$

L'action de la variable y *est définie par :*

$$\begin{aligned} A_y \triangleq\ & A_{dur_y} \\ & \wedge \mathit{Let}\ \langle \delta, \Delta \rangle = dur'_y\ \mathit{in} \begin{pmatrix} \hat{y}' = T' \wedge \mathtt{SporadicUpdate}(T, \hat{y}, \delta, \Delta) \\ \vee \\ \hat{y}' = \hat{y} \wedge \mathtt{SporadicIdle}(T, \hat{y}, \delta, \Delta) \end{pmatrix} \\ & \wedge \begin{pmatrix} \hat{y}' = T' \wedge \mathtt{PeriodicUpdate}(T, \hat{y}, P, J, \phi) \\ \vee \\ \hat{y}' = \hat{y} \wedge \mathtt{PeriodicIdle}(T, \hat{y}, P, J, \phi) \end{pmatrix} \end{aligned}$$

Ici, seule la périodicité et les conditions d'anticipation guident le comportement de la variable. On définit maintenant les actions des variables image d'une observation. On doit alors définir les actions construisant les conditions d'anticipation d'un chemin et les actions construisant les variables historiques.

Définition 74. *Condition d'anticipation d'un chemin. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et un chemin* $[x] : P : [z, y]$ *d'une variable* x *source vers une variable* y *et tel qu'il existe* Z *tel que* $z \in Z$ *et* $y \preccurlyeq f(Z)$*. On suppose que l'on a la propriété suivante sur le chemin* P*.*

$$P \left\{ \begin{array}{l} \mathtt{Latency}(\delta_{Lat}, \Delta_{Lat}), \mathtt{Lag}(\delta_{Lag}, \Delta_{Lag}), \mathtt{Shift}(\delta_{Shi}, \Delta_{Shi}), \\ \mathtt{Freshness}(\delta_{Fre}, \Delta_{Fre}), \mathtt{Fitness}(\delta_{Fit}, \Delta_{Fit}), \mathtt{Stability}(\delta_{Sta}, \Delta_{Sta}) \end{array} \right\}$$

L'action du chemin P *est définie par : :*

$$\begin{aligned} & A_P \triangleq dur'_P \triangleq \\ & \left\{ \begin{array}{l} \langle \max \begin{pmatrix} \delta_{Sta}, \delta_{Fre} + \delta_{Fit} + 1, \\ T' - upd + \delta_{Fit} - \Delta_{Shi} + 2 \end{pmatrix}, \min \begin{pmatrix} \Delta_{Sta} - 1, \Delta_{Fre} + \Delta_{Fit} - 1, \\ T' - upd + \Delta_{Fit} - \delta_{Shi} \end{pmatrix} \rangle \\ \mathit{si}\ \langle \hat{x}', T' + 1 - \hat{x}', [x] : P : [z, y] \rangle \in \min(H'_{y \preccurlyeq f(Z)}) \wedge \\ \quad \langle \hat{x}, T + 1 - \hat{x}, [x] : P : [z, y] \rangle \notin \min(H_{y \preccurlyeq f(Z)}) \\ \langle \max \begin{pmatrix} \delta, \delta_{Sta}, \delta_{Fre} + \delta_{Fit} + 1, \\ T' - upd + \delta_{Fit} - \Delta_{Shi} + 2 \end{pmatrix}, \min \begin{pmatrix} \delta, \Delta_{Sta} - 1, \Delta_{Fre} + \Delta_{Fit} - 1, \\ T' - upd + \Delta_{Fit} - \delta_{Shi} \end{pmatrix} \rangle \\ \mathit{si}\ \langle \hat{x}', T' + 1 - \hat{x}', [x] : P : [z, y] \rangle \in \min(H'_{y \preccurlyeq f(Z)}) \wedge \\ \quad \langle \hat{x}, T + 1 - \hat{x}, [x] : P : [z, y] \rangle \in \min(H_{y \preccurlyeq f(Z)}) \wedge \\ \quad dur_P = \langle \delta, \Delta \rangle \\ \langle 0, +\infty \rangle\ \mathit{sinon} \end{array} \right. \end{aligned}$$

Cette définition est une utilisation directe de la définition 70. Les conditions permettant de déterminer si une occurrence de la source est l'occurrence en cours est donné par l'appartenance des caractéristiques de cette occurrences au minimum de historique de l'observation définissant l'extrémité du chemin.

On définit maintenant une action donnant l'évolution d'une variable historique.

Définition 75. *Historique des occurrences. Pour une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$ *et une exécution* σ *satisfaisant cette spécification, pour une variable* y*, image d'une observation* $y \lesssim f(X)$*. L'action de l'historique d'une observation* $y \lesssim f(X)$ *est définie*[1] *par :*

$$
\begin{array}{rl}
A_{H_{y \lesssim f(X)}} \triangleq & \wedge\ \forall j \in [Index(min(H'_{y \lesssim f(X)}))..i] : \\
& \qquad H_{y \lesssim f(X)}[j]' = \left\{ o \;\middle|\; \begin{array}{l} \exists o' \in H_{y \lesssim f(X)}[j] : o' = \langle upd, dur, [z] : P\rangle \wedge \\ \left(\begin{array}{l} (\hat{z} \neq upd \ \wedge\ o = \langle upd, dur, [z] : P\rangle) \vee \\ (\hat{z} = upd \ \wedge\ o = \langle upd, T' + 1 - \hat{z}, [z] : P\rangle) \end{array} \right) \end{array} \right\} \\
& \wedge\ H_{y \lesssim f(X)}[IndexMax(H_{y \lesssim f(X)}) + 1]' = \{\langle \hat{x}', T' + 1 - \hat{x}', [x, y]\rangle | x \in X\} \cup \\
& \quad \left\{ o \;\middle|\; \begin{array}{l} x \in X \ \wedge\ \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H'_{x \lesssim g(Z)}) \\ \wedge\ o' = \langle upd, dur, P\rangle \ \wedge\ o = \langle upd, dur, P : [y]\rangle \ \wedge\ P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right\} \\
& \wedge\ \ \vee\ \hat{y}' = T' \ \wedge\ min(H'_{y \lesssim f(X)}) \neq min(H'_{y \lesssim f(X)}) \\
& \qquad \vee\ \hat{y}' = \hat{y} \ \wedge\ min(H'_{y \lesssim f(X)}) = min(H'_{y \lesssim f(X)}))
\end{array}
$$

Cette définition est une reformulation de la proposition 41. On ne connaît pas la taille de l'historique, cependant, le minimum de l'historique change si jamais la variable est mise à jour car les caractéristiques des occurrences utilisées change. On choisit l'indice minimum tel que l'élément à l'indice suivant ne correspond pas aux mêmes caractéristiques des occurrences. Il s'agit donc d'avoir cet indice minimum qui correspond bien à l'horloge maximum de l'observation.

On utilise ces deux actions pour définir l'action de l'image d'un chemin.

Définition 76. *Action d'une image sporadique. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$ *tel qu'il existe une observation* $y \lesssim f(X)$ *dans l'architecture du système. On suppose que la variable* y *est sporadique de paramètre* $\delta_{Spo}, \Delta_{Spo}$*. L'action de la variable* y *est définie par :*

$$
\begin{array}{l}
A_y \triangleq \\
\quad \wedge\ A_{H_{y \lesssim f(X)}} \ \wedge\ A_{dur_y} \\
\quad \wedge\ \textit{Let}\ \langle \delta, \Delta\rangle = dur'_y\ \textit{in} \\
\qquad\quad \vee\ \hat{y}' = T' \ \wedge\ SporadicUpdate(T, \hat{y}, min(\delta, \delta_{Spo}), max(\Delta, \Delta_{Spo})) \\
\qquad\quad \vee\ \hat{y}' = \hat{y} \ \wedge\ SporadicIdle(T, \hat{y}, min(\delta, \delta_{Spo}), max(\Delta, \Delta_{Spo})) \\
\quad \wedge\ \forall \langle upd, dur, [z] : P\rangle \in min(H'_{y \lesssim f(X)}) : \\
\qquad\quad \wedge\ A_{[z]:P} \\
\qquad\quad \wedge\ \ \vee\ (upd = 0 \ \wedge\ T' < \Delta_{[z]:P}) \\
\qquad\qquad \vee\ \ \wedge\ upd \neq 0 \\
\qquad\qquad\quad \wedge\ (upd \neq \hat{z}' \Rightarrow dur \in DurationInterval(T', upd, PathParameters([z] : P)) \\
\qquad\qquad\quad \wedge\ \ \vee\ \hat{y}' = T' \ \wedge\ T' \in \left(\begin{array}{c} PathInterval(upd, PathParameters([z] : P)) \cap \\ [upd + MinLag([z] : P)..upd + MaxLag([z] : P)[\end{array} \right) \\
\qquad\qquad\qquad\quad \vee\ \hat{y}' = \hat{y} \ \wedge\ T' \in PathInterval(upd, PathParameters([z] : P))
\end{array}
$$

Pour une telle variable, les conditions sont définies par la conjonction des conditions des propriétés sur les chemins, les conditions d'anticipation et éventuellement la sporadicité.

1 On reprend ici la notation du langage TLA. La notation :

$$\wedge\ A \\ \wedge\ B$$

est équivalente à $(A \wedge B)$ et la notation :

$$\vee\ A \\ \vee\ B$$

est équivalente à $(A \vee B)$

Définition 77. *Action d'une image périodique. Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$ tel qu'il existe une observation $y \preccurlyeq f(X)$ dans l'architecture du système et tel qu'il existe une propriété de périodicité sur cette variable :*

$$y\ \{\mathtt{Periodic}(P, J, \phi)\} \in \mathcal{P}rop$$

L'action de la variable y est définie par :

$$\begin{array}{l}
A_y \triangleq \\
\quad \wedge A_{H_{y \preccurlyeq f(X)}} \ \wedge A_{dur_y} \\
\quad \wedge \textit{ Let } \langle \delta, \Delta\rangle = dur'_y \textit{ in} \\
\qquad\quad \vee \hat{y}' = T' \ \wedge \ \mathtt{SporadicUpdate}(T, \hat{y}, \delta, \Delta) \\
\qquad\quad \vee \hat{y}' = \hat{y} \ \wedge \ \mathtt{SporadicIdle}(T, \hat{y}, \delta, \Delta) \\
\quad \wedge \ \ \vee \hat{y}' = T' \ \wedge \ \mathtt{PeriodicUpdate}(T, \hat{y}, P, J, \phi) \\
\qquad \vee \hat{y}' = \hat{y} \ \wedge \ \mathtt{PeriodicIdle}(T, \hat{y}, P, J, \phi) \\
\quad \wedge \forall \langle upd, dur, [z] : P\rangle \in \min(H'_{y \preccurlyeq f(X)}) : \\
\qquad\quad \wedge A_{[z]:P} \\
\qquad\quad \wedge \ \ \vee (upd = 0 \ \wedge \ T' < \Delta_{[z]:P}) \\
\qquad\qquad \vee \ \ \wedge upd \neq 0 \\
\qquad\qquad\quad \wedge (upd \neq \hat{z}' \Rightarrow dur \in \mathtt{DurationInterval}(T', upd, \mathtt{PathParameters}([z] : P)) \\
\qquad\qquad\quad \wedge \ \ \vee \hat{y}' = T' \ \wedge \ T' \in \left(\begin{array}{c} \mathtt{PathInterval}(upd, \mathtt{PathParameters}([z] : P)) \cap \\ [upd + \mathtt{MinLag}([z] : P)..upd + \mathtt{MaxLag}([z] : P)[\end{array} \right) \\
\qquad\qquad\qquad \vee \hat{y}' = \hat{y} \ \wedge \ T' \in \mathtt{PathInterval}(upd, \mathtt{PathParameters}([z] : P))
\end{array}$$

Pour une telle variable, les conditions sont définies par la conjonction des conditions de la propriété de périodicité, les conditions d'anticipation et les conditions des propriétés sur les chemins.

6.4.2 *État initial*

Les actions définies pour chaque variable définissent une relation de transition. Pour complètement définir un système de transition, on définit un ensemble d'états initiaux.

Définition 78. *État initial. Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$. On définit un ensemble d'états initiaux Σ_0 tel que :*

$$\begin{array}{ll}
\forall \sigma_0 \in \Sigma_0 : & \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{x}.\sigma_0 = 0 \ \wedge \ dur_x.\sigma_0 = \langle 0, +\infty\rangle \\
& \wedge \forall P \in \mathcal{P}aths(\mathcal{A}) : dur_P.\sigma_0 = \langle 0, +\infty\rangle \\
& \wedge \forall y \preccurlyeq f(X) \in \mathcal{A}rchi : H_{y \preccurlyeq f(X)}[0].\sigma_0 = \{\langle \hat{x}.\sigma_0, T.\sigma_0 + 1 - \hat{x}.\sigma_0, [x]\rangle | x \in X\} \cup \\
& \left\{ o \left| \begin{array}{l} x \in X \ \wedge \ \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in \min(H_{x \preccurlyeq g(Z)}.\sigma_0) \\ \wedge \ o' = \langle upd, dur, P\rangle \ \wedge \ o = \langle upd, dur, P : [x]\rangle \ \wedge \ P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right. \right\} \\
& \wedge \ \mathtt{IndexMin}(H_{y \preccurlyeq f(X)}.\sigma_0) = \mathtt{IndiceMax}(H_{y \preccurlyeq f(X)}.\sigma_0) = 0
\end{array}$$

6.4.3 *Système de transitions équivalent à une spécification*

Définition 79. *Étant donné une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$, on définit le système de transitions équivalent à cette spécification par un triplet $(\Sigma, \Sigma_0, \rightarrow)$ où l'ensemble des états initiaux Σ_0 est celui défini par la définition 78 et la relation de transition $\rightarrow$ est celle définie par la définition 62 page 103.*

On prouve que ce système de transitions définit bien un système de transitions équivalent à la spécification. On définit une équivalence entre état.

Définition 80. *Équivalence temporelle des états. Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$. Deux états σ_i et σ'_i de deux exécutions σ et σ' sont temporellement équivalentes par rapport à cette spécification si l'on a :*

$$\begin{aligned}\sigma_i \sim \sigma'_i \triangleq \quad & T.\sigma_i = T.\sigma'_i \ \wedge \ \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{\vec{x}}.\sigma_i = \hat{\vec{x}}.\sigma'_i \\ & \wedge \forall y \vec{<} f(X) \ \in \mathcal{A}rchi, max\{c(i) | c \in \mathcal{C}(y \vec{<} f(X)).\sigma\} = IndexMin(H_{y \vec{<} f(X)}.\sigma'_i)\end{aligned}$$

Deux états sont donc équivalents si les différentes variables du système ont été mises à jour au même moment.

On utilise cette relation entre les états de deux exécutions pour montrer que le système de transitions défini par un ensemble d'actions sur les variables définit un ensemble d'exécutions équivalents à l'ensemble des exécutions définis par la spécification. Pour cela, on prouve la propriété suivante :

Proposition 42. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et soit $(\Sigma, \Sigma_0, \rightarrow)$ le système de transitions défini par cette spécification. On démontre la propriété suivante :*

$$\begin{aligned}&\forall \sigma \in [\![\langle \mathcal{A}rchi, \mathcal{P}rop \rangle]\!]_{\mathcal{O}}, \exists \sigma' \in [\![(\Sigma, \Sigma_0, \rightarrow)]\!]_{ST} : \forall i \in \mathbb{N} : \sigma_i \sim \sigma'_i \\ &\qquad\qquad\qquad\qquad \wedge \\ &\forall \sigma' \in [\![(\Sigma, \Sigma_0, \rightarrow)]\!]_{ST}, \exists \sigma \in [\![\langle \mathcal{A}rchi, \mathcal{P}rop \rangle]\!]_{\mathcal{O}} : \forall i \in \mathbb{N} : \sigma_i \sim \sigma'_i\end{aligned}$$

Cette proposition indique donc que pour chaque exécution définie par la spécification, il existe une exécution équivalente définie par le système de transitions et qu'inversement pour chaque exécution du système de transition, il existe une exécution équivalente définie par la spécification. Les sémantiques des deux systèmes sont donc équivalentes. L'équivalence de deux exécutions est donné état par état.

Cette équivalence n'est pas une relation de bisimulation entre états. En effet, pour la spécification, pour un état, l'état postérieur dépend aussi des états antérieurs. L'équivalence est donc définie dans le cadre d'une exécution permettant alors d'utiliser les états antérieurs pour définir l'état suivant. La preuve est construite de manière similaire à la preuve d'une relation de bisimulation. La différence réside dans l'utilisation de l'équivalence des états antérieurs des exécutions des sémantiques des deux systèmes.

Démonstration. On démontre cette proposition en deux étapes.

1. On démontre tout d'abord que pour chaque exécution définie par la spécification, il existe une exécution équivalente définie par le système de transitions. On suppose que l'on a une exécution σ définie par la spécification. On prouve par récurrence sur les états qu'il existe une exécution σ' équivalente à σ et définie par le système de transition. On prouve donc que si pour un entier n, on peut pour tout k inférieur ou égal à n construire un état σ'_k équivalent à l'état σ_k et tel que $\sigma'_{k-1} \rightarrow \sigma'_k$, alors on pourra construire un état σ'_{n+1} équivalent à σ_{n+1} et tel que $\sigma'_n \rightarrow \sigma'_{n+1}$.

1.A On prouve tout d'abord la propriété au rang 0. Dans l'état σ_0, le temps, les profils temporels des différentes variables et les différentes horloges de l'observation sont égales à 0.

Un état initial du système de transitions est défini par :

$$\begin{array}{l}
\forall x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{\hat{x}}.\sigma'_0 = 0 \ \wedge \ dur_x.\sigma'_0 = \langle 0, +\infty \rangle \\
\wedge \ \forall P \in \mathcal{P}aths(\mathcal{A}) : dur_P.\sigma'_0 = \langle 0, +\infty \rangle \\
\wedge \ \forall y \lessdot f(X) \in \mathcal{A}rchi : H_{y \lessdot f(X)}[0].\sigma'_0 = \{\langle \hat{\hat{x}}.\sigma'_0, T.\sigma'_0 + 1 - \hat{\hat{x}}.\sigma'_0, [x] \rangle | x \in X\} \cup \\
\left\{ o \left| \begin{array}{l} x \in X \ \wedge \ \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \lessdot g(Z)}.\sigma'_0) \\ \wedge \ o' = \langle upd, dur, P \rangle \ \wedge \ o = \langle upd, dur, P : [x] \rangle \ \wedge \ \nexists P' : P = [y] : P' \end{array} \right. \right\} \\
\wedge \ IndexMin(H_{y \lessdot f(X)}.\sigma'_0) = IndiceMax(H_{y \lessdot f(X)}.\sigma'_0) = 0
\end{array}$$

On a bien :

$$\begin{array}{l}
T.\sigma_0 = T.\sigma'_0 \ \wedge \ \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{\hat{x}}.\sigma_0 = \hat{\hat{x}}.\sigma'_0 \\
\wedge \ \forall y \lessdot f(X) \ \in \mathcal{A}rchi : \exists \, c \in \mathcal{C}(obs).\sigma : c(0) = IndexMin(H_{y \lessdot f(X)}.\sigma'_0)
\end{array}$$

1.B. Soit un entier n. On suppose que pour tout k inférieur ou égal à n, on a construit un état σ'_k équivalent à un état σ_k. On construit un état σ'_{n+1} tel que $T.\sigma'_{n+1} = T.\sigma_{n+1}$ et équivalent à σ_{n+1}.

Indépendamment du fait qu'il y ait une mise à jour de y ou pas entre σ_n et σ_{n+1}, on prouve que la même évolution de y est autorisée par l'action associée à y entre σ'_n et σ'_{n+1}. L'évolution de y dépend des propriétés de la spécification sur y. On examine chacune des propriétés de la spécification déterminant si l'action A_y autorise le même comportement que la spécification.

Le respect des propriétés sur le comportement temporel de y dépend, d'après les proposition 35 36 de reformulation des propriétés de sporadicité et périodicité, de la valeur de $\hat{\hat{y}}$ et de T dans l'état précédent. Or les états σ_n et σ'_n sont équivalents et donc $\hat{\hat{y}}$ a la même valeur dans ces deux états. Si la spécification autorise une mise a jour, l'action de la variable y autorise la valeur de $\hat{\hat{y}}$ à être remplacée par la valeur de T. De même s'il n'y a pas mise à jour.

Si y est l'image d'une observation $y \lessdot f(X)$, il faut de plus que les propriétés des chemins dont y est l'image autorise cette mise à jour. Dans l'état σ'_{n+1}, on définit l'historique de l'observation en définissant son indice minimum comme égal à $c(n+1)$ où c est l' horloge de l'observation $y \lessdot f(X)$ ayant la valeur maximum en $n+1$. On prouve que ce choix est autorisé par l'action A_y :

- L'indice minimum de l'historique choisi est supérieur à l'indice minimum de l'historique dans l'état précédent. En effet, toutes les horloges de l'observation croient entre σ_n et σ_{n+1} et donc l'indice choisi en σ'_{n+1} est nécessairement plus élevé que l'indice minimum choisi en σ'_n. D'après la définition de A_y, en dehors de l'élément à l'indice maximum, l'historique en σ'_{n+1} est défini par les éléments de l'historique dans l'état σ'_n. L'indice minimum croissant, on utilise bien des éléments qui existent dans l'historique en l'état σ'_n. L'historique peut donc bien être construit.
- Les états antérieurs de l'exécution du système de transitions sont équivalents aux états de l'exécution de la spécification et donc les occurrences de chacune des variables des états antérieurs ont les mêmes caractéristiques dans chacune des exécutions. La définition de l'action 75 construisant l'historique vérifie la proposition 41 et construit donc l'historique des caractéristiques des occurrences des sources dans les états antérieurs.
- Le minimum de l'historique contient donc les caractéristiques des occurrences sources utilisée pour définir l'occurrence de l'image. L'action A_y vérifie pour chaque élément du minimum de l'historique que le temps T en σ'_{n+1} appartient à l'intervalle $PathInterval$. D'après la proposition 38 l'action A_y vérifie cette propriétés, ces mêmes caractéristiques vérifiant les propriétés de l'exécution entre σ_n

et σ_{n+1}. On utilise donc les mêmes occurrences des sources pour définir l'image d'une observation.
- Finalement, on vérifie les propriétés sur la durée des occurrences afin de compléter la reformulation des propriétés d'un chemin donné par la proposition 38. Si pour la source d'un chemin, l'occurrence du minimum de l'historique n'est pas l'occurrence en cours alors on vérifie l'appartenance à un intervalle tel que défini par la proposition 38. Si jamais, il s'agit de l'occurrence en cours, alors, c'est la construction de la condition d'anticipation qui fait que la durée devra dans les états successeurs appartenir à cette intervalle. Dans l'exécution σ, cette durée doit satisfaire l'appartenance au même intervalle. On est donc assuré que dans cette exécution σ, à l'instant où cette source sera mise à jour, on pourra construire dans l'état σ' un état équivalent où cette source est aussi mise à jour.

On déduit alors que dans ce cas, on a bien la valeur du profil temporel de y qui est la même dans les deux états et qu'il existe bien une horloge de l'observation définissant l'indice minimum de l'historique. Ceci est vrai pour toutes les variables et toutes les observations, on construit donc bien un état σ'_{n+1} équivalent à l'état σ_{n+1}.

2. Réciproquement, on démontre de manière similaire que pour chaque exécution définie par le système de transitions, il existe une exécution équivalente définie par la spécification. On suppose que l'on a une exécution σ' définie par le système de transitions. On prouve par récurrence sur les états qu'il existe une exécution σ équivalente à σ' et définie par la spécification. On prouve donc que si pour un entier n, on peut pour tout k inférieur ou égal à n construire un état σ_k équivalent à l'état σ'_k, alors on pourra construire un état σ_{n+1} équivalent à σ'_{n+1}.

Pour prouver cette équivalence, on utilise le fait que l'architecture est satisfiable et donc qu'il existe des valeurs des variables vérifiant cette architecture. On fixe un ensemble de valeurs vérifiant l'architecture dans l'état initial. Par la suite, dans tout état, les variables gardent cette même valeur. Pour chaque variable non image d'une observation, on considère alors qu'on a une mise à jour ne donnant pas une nouvelle valeur. Pour chaque variable image d'une observation, les mises à jour se font alors avec de nouvelles occurrences des sources mais ayant la même valeur. Cela nous permet de construire simplement les valeurs des variables du système.

2.A. La propriété de rang 0 est prouvée en donnant aux différentes variables du système une valeur vérifiant les relations d'observations de l'architecture. Comme pour la première partie de la démonstration, les valeurs du temps, des profils temporels et de l'historique définissent deux états équivalents.

2.B. Soit un entier n. On suppose que pour tout k inférieur ou égal à n, on a construit un état σ_k équivalent à un état σ'_k. On construit un état σ_{n+1} équivalent à σ_{n+1}. On définit un état σ_{n+1} tel que $T.\sigma_{n+1} \neq T.\sigma_n$.

Indépendamment du fait qu'il y ait eu une mise à jour de y entre σ'_n et σ'_{n+1}, mise à jour indiquée par un changement de valeur de $\hat{y}$, on prouve que la même évolution de y est autorisée par la spécification. On examine chacune des propriétés de la spécification déterminant s'il peut y avoir une mise à jour ou non.

L'évolution de y dépend des propriétés sur le comportement temporel de y. Le respect des propriétés sur le comportement temporel de y, sporadicité et périodicité, dépend d'après les propositions 36 et 35 de la valeur de $\hat{y}$ et de T dans l'état précédent. Or les états σ_n et σ'_n sont équivalents et donc $\hat{y}$ a la même valeur dans ces deux états. Si

l'action de la variable y autorise une mise a jour, la spécification autorise une mise à jour et la valeur de $\hat{y}$ a être remplacé par la valeur de T et donc à indiquer un changement d'occurrence. On garde cependant la même valeur de y. De même s'il n'y a pas mise à jour.

Si y est l'image d'une observation $y \preccurlyeq f(X)$, il faut de plus que les propriétés des chemins dont y est l'image autorise cette mise à jour. On a vu dans la proposition 38 qu'il suffisait que les caractéristiques des occurrences passées des sources des chemins vérifient certaines conditions pour qu'on puisse construire pour chaque chemin une horloge satisfaisant les propriétés de la spécification. Or on sait que :

- Les états antérieurs de l'exécution du système de transitions sont équivalents aux états de l'exécution de la spécification et donc les occurrences de chacune des variables des états antérieurs ont les mêmes caractéristiques dans chacune des exécutions. L'action de l'historique donnée définition 75 construisant l'historique vérifie la proposition 41 et construit donc l'historique des caractéristiques des occurrences des sources dans les états antérieurs. Les états antérieurs des deux exécutions sont équivalents donc l'historique contient les caractéristiques des occurrences des sources des états antérieurs à σ'_{n+1} mais contient donc aussi les caractéristiques des occurrences des sources des états antérieurs à σ_{n+1}.
- Le minimum de l'historique donne un ensemble de caractéristiques qui vérifie la spécification d'après les reformulations des propriétés et la définition des actions associées aux variables. Ces caractéristiques correspondent aux occurrences des sources dans les états antérieurs σ_{n+1} et vérifient les propriétés de la spécification. Pour chaque chemin, il existe donc une horloge liant l'état courant à un état antérieur à σ_{n+1} et vérifiant les propriétés de ce chemins.
- Selon que le minimum de l'historique diffère de l'état précédent, il y a ou pas changement de valeur de $\hat{y}$ et mise à jour. S'il y a mise à jour, on met à jour $\hat{y}$ sans modifier la valeur de y.

On déduit alors que dans ce cas, on a bien la valeur du profil temporel de y qui est la même dans les deux états et qu'il existe bien une horloge de l'observation définissant l'indice minimum de l'historique. Ceci est vrai pour toutes les variables et toutes les observations, on construit donc bien un état σ_{n+1} équivalent à l'état σ'_{n+1}. □

7 ÉQUIVALENCE À UN SYSTÈME FINI

Dans le chapitre précédent, on a construit un modèle opérationnel équivalent à une spécification à base d'observations. Ce modèle, sous forme de système de transitions, permet de construire des exécutions équivalentes à celles vérifiant cette spécification. Cependant, ce système de transitions n'est pas un système d'états finis.

On définit ici un système de transitions équivalent au système de transitions définis par une spécification mais définissant un système d'états finis. La définition de ce système repose sur une réduction du domaine de valeurs des variables.

7.1 INTRODUCTION DU PROBLÈME ET DE SA SOLUTION

7.1.1 *Introduction*

Le système de transitions défini dans le chapitre précédent est infini du fait que le temps T et les variables qui lui sont liés sont croissantes non bornées. Il n'est donc pas possible d'utiliser ce système pour dérouler directement une exécution infinie et construire tous les états de cette exécution. On construit un système de transitions fini et équivalent au système de transitions défini dans le chapitre précédent.

Les propriétés temporelles d'une spécification sont définies par des bornes sur les différences entre les différentes valeurs des variables du système, notamment les profils temporels des variables, le temps T et la durée des occurrences d'une variable. Le comportement du système est donc défini par ces différences et le comportement du système de transitions équivalent à une spécification est lui aussi défini par ces différences. Deux états pour lesquelles ces différences sont égales ont donc les mêmes successeurs les mêmes propriétés étant satisfaites. On définit donc une relation d'équivalence entre états qui préserve ces différences.

7.1.2 *Définition de la relation d'équivalence*

La relation d'équivalence est définie en utilisant l'opération modulo. Pour un entier naturel L, on utilise la définition suivante pour donner pour tout entier relatif k le résultat de "k modulo L" noté $k \pmod L$:

$$\forall k \in \mathbb{Z} : (\exists n \in \mathbb{Z} : k = n * L + (k \pmod L)) \wedge k \pmod L \in [0..L[$$

Les propriétés de la division euclidienne prouvent l'existence et l'unicité de cette valeur pour $L > 0$.

Définition 81. *Equivalence modulo* L *de deux états. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$. *Deux états de ce système sont équivalents modulo* L *si on a :*

$$
\begin{aligned}
\sigma_i \underset{L}{\sim} \sigma_j \triangleq \quad & T.\sigma_i \pmod L = T.\sigma_j \pmod L \\
& \wedge \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{\hat{x}}.\sigma_i \pmod L = \hat{\hat{x}}.\sigma_i \pmod L \\
& \wedge \forall obs \in \mathcal{A}rchi, \forall k \in [IndexMin(H_{obs}.\sigma_i)..IndexMax(H_{obs}.\sigma_i)], \\
& \forall o \in H_{obs}[k].\sigma_i, \exists o' \in H_{obs}[k].\sigma_j : \\
& o = \langle upd, dur, P \rangle \wedge o' = \langle upd', dur, P \rangle \wedge upd \pmod L = upd' \pmod L \\
& \wedge \forall obs \in \mathcal{A}rchi, \forall k \in [IndexMin(H_{obs}.\sigma_j)..IndexMax(H_{obs}.\sigma_j)], \\
& \forall o' \in H_{obs}[k].\sigma_j, \exists o \in H_{obs}[k].\sigma_i : \\
& o' = \langle upd', dur, P \rangle \wedge o = \langle upd, dur, P \rangle \wedge upd \pmod L = upd' \pmod L
\end{aligned}
$$

Deux états σ_i et σ_j sont donc équivalents modulo L si quatre propriétés sont respectées :

- les valeurs de T dans les deux états sont égales modulo L ;
- les valeurs des profils temporels de chaque variable dans les deux états sont égales modulo L ;
- pour chaque triplet de caractéristiques d'une occurrence de l'historique d'une observation dans l'état σ_i, l'historique de la même observation dans l'état σ_j contient un triplet caractéristique d'une occurrence du même chemin, avec la même durée et la même date de mise à jour modulo L et rangé au même indice ;
- et inversement entre σ_j et σ_i

Pour résumé, les deux états sont définis par les mêmes caractéristiques du comportement des variables et les mêmes variables historique à l'exception que les variables prenant valeurs dans le domaine de définition de T sont égales modulo L.

La définition de cette relation d'équivalence entre états est motivée par la propriété suivante :

Lemme 43. *Soit un entier positif non nul* L. *Pour deux entiers* x *et* y *; on a*

$$0 \leq x - y < L \Leftrightarrow x - y = ((x \pmod L) - y \pmod L) \pmod L$$

Démonstration. Soit un entier non nul L et deux entiers x et y. On prouve l'implication dans les deux sens. Dans les deux cas, on utilise la division euclidienne et l'existence de quatre entiers d_x, d_y, r_x, r_y tel que r_x et r_y sont dans l'intervalle $[0..L[$ et :

$$x = d_x * L + r_x \wedge y = d_y * L + r_y$$

On prouve d'abord l'implication de gauche à droite.

1. On suppose que l'on a :

$$0 \leq x - y < L$$

On a :

$$x - y = d_x * L + r_x - d_y * L - r_y = (d_x - d_y)L + r_x - r_y$$

Or d'après les définitions de r_x et r_y, on a $-L < r_x - r_y < L$. On considère deux cas selon le signe de cette différence.

1.A. On suppose tout d'abord que l'on a $-L < r_x - r_y < 0$. Alors on $0 < r_y - r_x < L$.

$$
\begin{array}{ll}
 & 0 \leq x - y < L \\
\Rightarrow & \quad \{\text{réécriture}\} \\
 & 0 \leq (d_x - d_y)L + r_x - r_y < L \\
\Rightarrow & \quad \{\text{ajout de } 0 < r_y - r_x < L\} \\
 & 0 < (d_x - d_y)L < 2L \\
\Rightarrow & \quad \{\text{simplification par } L\} \\
 & 0 < d_x - d_y < 2
\end{array}
$$

On a donc $d_x - d_y = 1$ et donc $x - y = L + r_x - r_y$. Dans ce cas :

$$((x \pmod L) - y \pmod L) \pmod L) = (r_x - r_y) \pmod L = (r_x - r_y) + L$$

Car $r_x - r_y \in]-L..0[$. On a donc bien $x - y = ((x \pmod L) - y \pmod L) \pmod L)$.

1.B. On suppose que l'on a $0 \leq r_x - r_y < L$. On a alors $-L < r_y - r_x \leq 0$.

$$
\begin{array}{ll}
 & 0 \leq x - y < L \\
\Rightarrow & \quad \{\text{réécriture}\} \\
 & 0 \leq (d_x - d_y)L + r_x - r_y < L \\
\Rightarrow & \quad \{\text{ajout de } -L < r_y - r_x \leq L\} \\
 & -L < (d_x - d_y)L < L \\
\Rightarrow & \quad \{\text{simplification par } L\} \\
 & -1 < d_x - d_y < 1
\end{array}
$$

On a donc $d_x - d_y = 0$ et donc $x - y = r_x - r_y$. Dans ce cas :

$$((x \pmod L) - y \pmod L) \pmod L) = (r_x - r_y) \pmod L = (r_x - r_y)$$

Car $r_x - r_y \in [0..L[$. On a donc bien $x - y = ((x \pmod L) - y \pmod L) \pmod L)$.

2. Voyons maintenant la réciproque. On suppose que l'on a :

$$x - y = ((x \pmod L) - y \pmod L) \pmod L)$$

Alors, par définition on sait que l'on a :

$$0 \leq ((x \pmod L) - y \pmod L) \pmod L) < L$$

On a donc bien $0 \leq x - y < L$. □

Les propriétés de la spécification sont basées sur les différences entre les valeurs des variables temps, profil temporel et stabilité prises sur une séquence d'états. Si on connaît une valeur qui borne ces différences, on peut utiliser la valeur de ces variables modulo cette borne au lieu de leur valeur effective. L'utilisation du reste de la division euclidienne au lieu de la valeur des variables préserve le comportement du système tout en limitant le domaine de définition des variables et donc le nombre d'états possibles. On prouve que pour une spécification il existe bien un entier L bornant les différences entre variables utilisées pour définir les propriétés de la spécification.

7.1.3 *Choix de l'intervalle d'analyse*

D'après la proposition 27 page 75, il est possible d'ordonner en tout état les valeurs des différentes variables utilisées pour définir les propriétés d'un chemin. On a alors les propositions suivantes :

Lemme 44. *Soit un ensemble d'observation d'occurrences* $\bar{O}bs$*. Pour un chemin* P *bien défini par* $\bar{O}bs$ *d'une variable* x *vers une variable* y*, on a :*

$$\begin{aligned}\forall c \in \mathbb{N}, \forall i \in \mathbb{N} : \quad & 0 \leq T.\sigma_i - T.\sigma_{c(i)} \leq T.\sigma_i - \hat{\hat{x}}.\sigma_{c(i)} \\ & 0 \leq T.\sigma_{c(i)} - \hat{\hat{x}}.\sigma_{c(i)} \leq T.\sigma_i - \hat{\hat{x}}.\sigma_{c(i)} \\ & 0 \leq \hat{\hat{y}}.\sigma_i - \hat{\hat{x}}.\sigma_{c(i)} \leq T.\sigma_i - \hat{\hat{x}}.\sigma_{c(i)} \\ & 0 \leq \hat{\hat{x}}.\sigma_{c(i)} + d_{\bar{x}}.\sigma_{c(i)} \leq d_{\bar{x}}.\sigma_{c(i)}\end{aligned}$$

Justification. Cette proposition est une implication directe de la proposition 27

Les différences qui sont utilisées pour définir les propriétés de la spécification sont donc bornées par la différence définissant la latence et la durée des occurrences des variables. D'après le lemme 43, il est donc possible de calculer ces différences entre deux valeurs à partir des valeurs modulo les bornes supérieures sur la latence et la valeur de la stabilité de la source. On veut donc trouver une borne sur la latence des chemins et la stabilité des variables de l'architecture.

Proposition 45. *Soit un ensemble d'observation d'occurrences* $\bar{O}bs$*. Pour un chemin* P *bien défini par* $\bar{O}bs$ *d'une variable* x *vers une variable* y*, on a :*

$$\sigma \vDash x\ \{Sporadic(0, \Delta_x)\} \ \wedge\ \Delta_y = \Delta_P \Rightarrow\ \sigma \vDash y\ \{Sporadic(0, \Delta_y)\}$$

avec Δ_x*,* Δ_y *les temps de sortie du régime initial de* x *et* y *et* Δ_P *le temps de sortie du régime initial du chemin* P*.*

Démonstration. Soit un chemin P entre deux variables x et y tel que x n'est pas image d'une observation. Alors, pour tout entier i et pour une horloge c du chemin P vérifiant les propriétés de latence et de décalage, on a :

$$(T.\sigma_i - T.\sigma_{c(i)} < MaxShift(P) \wedge T.\sigma_i - \hat{\hat{x}}.\sigma_{c(i)} < MaxLatency(P)) \vee (T.\sigma_i < \Delta_P \wedge \hat{\hat{x}}.\sigma_{c(i)} = 0)$$

Comme x est sporadique, alors on a par définition :

$$\forall i \in \mathbb{N} : d_{\bar{x}}.\sigma_i \leq \Delta_x \ \vee\ (T.\sigma_i < \Delta_x \ \wedge\ \hat{\hat{x}}.\sigma_i = 0)$$

Pour prouver que y est sporadique, on utilise un raisonnement par l'absurde. Alors il existe un entier i tel que :

$$d_{\bar{y}}.\sigma_i > \Delta_P \ \wedge\ (T.\sigma_i \geq \Delta_y \ \vee\ \hat{\hat{y}}.\sigma_i \neq 0)$$

On considère deux cas selon que P est sorti de son régime initial ou pas.

1. On suppose que P est dans son régime initial. On a alors $T.\sigma_i < \Delta_P$. Par définition de l'état i, et comme $T.\sigma_i < \Delta_P$, on a alors $\hat{\hat{y}}.\sigma_i \neq 0$. Or P doit sortir du régime initial avant l'instant Δ_P. Cette sortie du régime initial entraînera une mise à jour de y au pire à l'instant Δ_P. Comme $\hat{\hat{y}}.\sigma_i > 0$, le temps entre ces deux mises à jour ne peut être supérieur à Δ_P ce qui est en contradiction avec $d_{\bar{y}}.\sigma_i > \Delta_P$. On en déduit que dans l'état i, P n'est pas dans le régime initial.

2. On suppose désormais que dans l'état σ_i, P n'est pas dans le régime initial. La durée de l'occurrence de y dans l'état i est supérieur à Δ_P. Le dernier état j de l'occurrence en i est tel que $j \geq i$ et $\hat{\hat{y}}.\sigma_j = \hat{\hat{y}}.\sigma_i$ et :

$$T.\sigma_j - \hat{\hat{y}}.\sigma_j \geq \Delta_P$$

$j \geq i$ et donc P est sorti du régime initial dans l'état σ_j. D'après la proposition 27, on a pour toute horloge c du chemin P :

$$\begin{array}{ll} & \hat{\hat{y}}.\sigma_j - \hat{\hat{x}}.\sigma_{c(j)} \geq 0 \\ \Rightarrow & \quad \{\text{ajout de } T.\sigma_j - \hat{\hat{y}}.\sigma_j \geq \Delta_P\} \\ & T.\sigma_j - \hat{\hat{x}}.\sigma_{c(j)} \geq \Delta_P \end{array}$$

D'après le choix 4 :

$$\Delta_P = \Delta_x + \min(MaxLatency(P), MaxShift(P))$$

On distingue deux cas selon la définition de Δ_P.

2.A. On suppose que l'on a :

$$\Delta_P = \Delta_x + MaxLatency(P))$$

Alors :

$$T.\sigma_j - \hat{\hat{x}}.\sigma_{c(j)} \geq \Delta_x + MaxLatency(P))$$

L'inégalité de la propriété de latence n'est donc pas vérifiée alors que le chemin P est sorti du régime initial. On a donc une contradiction et l'hypothèse de départ est fausse, donc y est sporadique.

2.B. On suppose que l'on a :

$$\Delta_P = \Delta_x + MaxShift(P)$$

On a la propriété de décalage et donc on a une horloge c du chemin P tel que :

$$T.\sigma_j - T.\sigma_{c(j)} < MaxShift(P)$$

Comme x est sporadique, on a de plus nécessairement :

$$\begin{array}{ll} & T.\sigma_{c(j)} - \hat{\hat{x}}.\sigma_{c(j)} < \Delta_x \\ \Rightarrow & \quad \{\text{ajout de } T.\sigma_j - T.\sigma_{c(j)} < MaxShift(P)\} \\ & T.\sigma_j - \hat{\hat{x}}.\sigma_{c(j)} < \Delta_x + MaxShift(P) \end{array}$$

D'après la proposition 27, on a :

$$\begin{array}{ll} & \hat{\hat{x}}.\sigma_{c(j)} < \hat{\hat{y}}.\sigma_j \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & \hat{\hat{x}}.\sigma_{c(j)} - \hat{\hat{y}}.\sigma_j < 0 \\ \Rightarrow & \quad \{\text{ajout de } T.\sigma_j - \hat{\hat{x}}.\sigma_{c(j)} < \Delta_x + MaxShift(P)\} \\ & T.\sigma_j - \hat{\hat{y}}.\sigma_{c(j)} < \Delta_x + MaxShift(P) = \Delta_P \end{array}$$

On avait supposé que $T.\sigma_j - \hat{\hat{y}}.\sigma_j \geq \Delta_P$, on a donc une contradiction et l'hypothèse de départ est fausse, donc y est sporadique. □

Proposition 46. *Étant donné un ensemble d'observations d'occurrence $\bar{\mathcal{O}}bs$, deux variables y et x et un chemin P de x vers y bien défini par rapport à $\bar{\mathcal{O}}bs$, alors pour une exécution σ on a :*

$$\sigma \vDash \forall \Delta_1, \Delta_2 \in \mathbb{N} : P\,\{Shift(0,\Delta_1)\} \wedge x\,\{Sporadic(0,\Delta_2)\} \Rightarrow (P\,\{Latency(0,\Delta_1+\Delta_2)\})$$

Démonstration. Soit un ensemble d'observations $\bar{\mathcal{O}}bs$, deux variables y et x et un chemin P de x vers y bien défini par rapport à $\bar{\mathcal{O}}bs$. Alors pour une exécution σ, on suppose que l'on a :

$$P\,\{Shift(0,\Delta_1)\} \wedge x\,\{Sporadic(0,\Delta_2)\}$$

Pour un état σ_i du régime initial du chemin P, on a bien la propriété de latence P étant dans le régime initial.

Soit une horloge c du chemin vérifiant la propriété de décalage et un état i où P est sorti du régime initial. On a donc $\hat{\bar{x}}.\sigma_{c(i)} \neq 0$. On a donc nécessairement x qui est sorti du régime initial en $c(i)$. Donc on a :

$$T.\sigma_i - T.\sigma_{c(i)} \leq \Delta_1 \ \wedge\ d_{\bar{x}}.\sigma_{c(i)} \leq \Delta_2$$

En $c(i)$, on a : $T.\sigma_{c(i)} \leq d_{\bar{x}}.\sigma_{c(i)} + \hat{\bar{x}}.\sigma_{c(i)}$ d'où :

$$\begin{array}{ll} & T.\sigma_i - T.\sigma_{c(i)} \leq \Delta_1 \ \wedge\ T.\sigma_{c(i)} - \hat{\bar{x}}.\sigma_{c(i)} \leq \Delta_2 \\ \Rightarrow & \quad \{\text{addition des deux inégalités}\} \\ & T.\sigma_i - \hat{\bar{x}}.\sigma_{c(i)} \leq \Delta_1 + \Delta_2 \end{array}$$

Et donc on a bien la propriété de latence. □

Définition 82. *Longueur de l'intervalle d'analyse d'une spécification. La longueur de l'intervalle d'analyse d'une spécification $S = \langle \mathcal{A}rchi, \mathcal{P}rops \rangle$ est un entier L_S tel que pour toute exécution σ vérifiant la spécification :*

$$\begin{array}{l} \forall P \in \mathcal{P}aths(\mathcal{A}rchi) : \\ \sigma \vDash P\,\{Latency(0, L_S - 1)\} \ \wedge\ \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : \sigma \vDash x\,\{Sporadic(0, L_S - 1)\} \end{array}$$

Justification. Une telle durée borne donc la latence de tous les chemins de l'architecture ainsi que la sporadicité de toute les variables.

Dans la définition des actions de la relation de transition, on est amené à vérifier si cette valeur maximale est atteinte. On choisit donc d'avoir une valeur qui est supérieure de un à la borne maximum de la sporadicité et de la latence afin de ne jamais comparer deux valeurs dont la différence atteint cette valeur. Par exemple, pour une variable x et un état σ_i, on se base sur la valeur $T.\sigma_i - \hat{\bar{x}}.\sigma_{i-1} = T.\sigma_{i-1} + 1 - \hat{\bar{x}}.\sigma_{i-1}$ pour décider de mettre à jour x. Cette valeur est susceptible d'atteindre la borne supérieure de la sporadicité. Comme on ne veut jamais que la différence qu'on considère atteigne L_S, on choisit donc comme valeur la borne supérieure de la sporadicité augmentée de 1.

Choix 6. *Soit une spécification $S = \langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ vérifiant la proposition 34 page 98. On choisit alors comme longueur L_S de l'intervalle d'analyse le premier entier :*

- *multiple commun des périodes des variables ;*
- *strictement supérieur au temps de sortie du régime initial de cette spécification.*

Justification. Pour justifier ce choix, on prouve que le temps de sortie du régime initial d'une spécification vérifie bien la définition de la longueur de l'intervalle d'analyse. On

considère une spécification vérifiant la proposition 34 page 98 et assurant donc que le temps de sortie du régime initial de cette spécification est borné.

D'après la proposition 33, toute variable x de l'architecture a un temps de sortie du régime initial définit par le temps de sortie du régime initial d'un chemin. Ce chemin a pour image x et pour source une variable non image d'une observation. Cette source du chemin, d'après la proposition 34, est soit périodique, soit sporadique avec une borne maximum sur cette propriété de sporadicité. La périodicité implique la sporadicité. On a donc bien une variable sporadique source d'un chemin dont le temps de sortie du régime initial est le temps de sortie du régime initial de l'image. D'après la proposition 45, l'image est donc sporadique avec comme borne supérieure le temps de sortie du régime initial d'un chemin.

Pour tous les chemins d'une spécification avec un temps de sortie du régime initial bornée, il existe soit un prédicat de latence avec une borne supérieure, soit un prédicat de décalage avec une borne supérieure ou alors il existe un prédicat de latence avec une borne supérieure sur un sur-chemin. On considère chacun de ces cas.

Par définition du temps de sortie du régime initial, si jamais un chemin est soumis à un prédicat de latence, alors le temps de sortie du régime initial du chemin et donc de la spécification est supérieur à la borne supérieure de ce prédicat.

Si jamais ce chemin a uniquement un prédicat de décalage qui est spécifié, la proposition 46 nous dit qu'il respecte cependant un prédicat de latence avec une borne supérieure égale à la somme de la borne supérieure du décalage et la borne supérieure de la sporadicité de la source du chemin. Or d'après la proposition 45, toute variable est sporadique et la borne supérieure de la sporadicité de la source du chemin est inférieure à la borne donnée par la propriété 45 qui est le temps de sortie du régime initial de cette source. La somme de ces deux bornes correspond alors au temps de sortie du régime initial du chemin.

Finalement, si le temps de sortie du régime initial d'un chemin P est défini par rapport à la latence d'un sur-chemin P', alors la latence de P est borné par la latence de P' d'après la propriété 30, latence elle même bornée par le temps de sortie du régime initial de la spécification. La latence de chaque chemin est donc bornée par le temps de sortie du régime initial de la spécification.

Le choix d'un entier multiple commun des périodes des variables permet de satisfaire plus simplement la périodicité de ces variables dans le système de transitions réduit défini par la suite.

7.2 SYSTÈME DE TRANSITIONS FINI ÉQUIVALENT À UNE SPÉCIFICATION

7.2.1 *Relation de transition*

Pour définir le système de transitions fini équivalent à une spécification, on utilise le même état initial que dans le système de transitions défini dans le chapitre précédent. On définit par contre une nouvelle relation de transition modifiant l'écoulement du temps et redéfinissant les actions associées à chaque variable.

Définition 83. *Relation de transition globale. Étant donné une spécification* $\langle \mathcal{A}\mathrm{rchi}, \mathcal{P}\mathrm{rop} \rangle$, L_S *la longueur de l'intervalle d'analyse de cette spécification et l'ensemble des variables du*

système $\mathcal{V}ar(\mathcal{A}rchi)) = \{x_k | k \in [1..n]\}$*, on définit la relation de transition globale* $\rightarrow_{L_S}$ *de cette spécification par :*

$$\sigma_i \rightarrow_{L_S} \sigma_{i+1} \vDash \bigwedge_{k=1}^{n} A_{x_k}.\sigma_i.\sigma_{i+1} \ \wedge \ \left(\begin{array}{l} T.\sigma_{i+1} = T.\sigma_i + 1 - L_S \textit{ si } T.\sigma_i + 1 \geq 2 * L_S \\ T.\sigma_{i+1} = T.\sigma_i + 1 \textit{ sinon} \end{array} \right)$$

et où chaque A_{x_k} *est une action, un prédicat portant sur deux états, définie à partir de la spécification et indiquant si la variable* x_k *peut être mise à jour et avec quelle occurrence.*

Si T est initialisé à 0 alors les valeurs de T sont donc, en dehors des L_S premiers états d'une exécution, comprises dans l'intervalle $[L_S..2 * L_S[$. Les L_S premiers instants correspondent au régime initial d'une exécution. Les valeurs du profil temporel étant définies par les valeurs de T, on est alors assuré de ne pas donner au profil temporel la valeur 0 qui est utilisée pour définir le régime initial en dehors de ce régime initial.

7.2.2 *Action associée à une variable*

On utilise les valeurs des variables modulo la longueur de l'analyse d'intervalle. La spécification définit des exécutions où la valeur du profil temporel d'une variable est en tout état inférieur à T. Cependant, les valeurs modulo L ne respectent pas cette inégalité. On donne la fonction suivante qui permet de retrouver une valeur du profil temporel inférieure à T.

Définition 84. *On définit la fonction* ValMod *pour trois entiers :*

$$ValMod(upd, T, L_S) \triangleq -((T - upd) \pmod{L_S}) + T$$

On redéfinit alors les différentes actions du chapitre précédent section 7.2.2 page 146. Les actions sont modifiées par l'utilisation de la fonction ValMod appliquée au valeur représentant une date de mise à jour au lieu d'utiliser directement cette date de mise à jour.

Définition 85. *Condition d'anticipation d'une variable. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification. Pour une variable* x*, on définit l'action sur la condition d'anticipation de* x *par :*

$$A_{dur_x} \triangleq dur'_x = \left\langle \begin{array}{l} max\{\delta_{[x]:P} | \exists [x] : P \in \mathcal{P}aths(\mathcal{A}rchi) : dur'_{[x]:P} = \langle \delta_{[x]:P}, \Delta_{[x]:P} \rangle\}, \\ min\{\Delta_{[x]:P} | \exists [x] : P \in \mathcal{P}aths(\mathcal{A}rchi) : dur'_{[x]:P} = \langle \Delta_{[x]:P}, \Delta_{[x]:P} \rangle\} \end{array} \right\rangle$$

Définition 86. *Action d'une variable sporadique. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification, tel qu'il n'existe pas d'observation* $y \lessdot f(X)$ *dans l'architecture du système. On suppose que la variable est sporadique de paramètres* $\delta_{Spo}, \Delta_{Spo}$*. L'action de la variable* y *est définie par :*

$$A_y \triangleq A_{dur_y} \ \wedge \textit{ Let } \langle \delta, \Delta \rangle = dur'_y \textit{ in}$$
$$\left(\begin{array}{c} \hat{y}' = T' \ \wedge \ SporadicUpdate(T, ValMod(\hat{y}, T, L_S), min(\delta, \delta_{Spo}), max(\Delta, \Delta_{Spo})) \\ \vee \\ \hat{y}' = \hat{y} \ \wedge \ SporadicIdle(T, ValMod(\hat{y}, T, L_S), min(\delta, \delta_{Spo}), max(\Delta, \Delta_{Spo})) \end{array} \right)$$

Définition 87. *Action d'une variable périodique. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification, tel qu'il n'existe pas d'observation*

$y \preccurlyeq f(X)$ *dans l'architecture du système et tel qu'il existe une propriété de périodicité sur cette variable :*

$$y\ \{\mathrm{Periodic}(P, J, \phi)\} \in \mathcal{P}\mathrm{rop}$$

L'action de la variable y *est définie par :*

$$\begin{aligned} A_y \triangleq\ & A_{dur_y} \\ & \wedge \textit{ Let } \langle \delta, \Delta \rangle = dur'_y \textit{ in } \left(\begin{array}{c} \hat{y}' = T' \wedge \mathrm{SporadicUpdate}(T, \mathrm{ValMod}(\hat{y}, T, L_S), \delta, \Delta) \\ \vee \\ \hat{y}' = \hat{y} \wedge \mathrm{SporadicIdle}(T, \mathrm{ValMod}(\hat{y}, T, L_S), \delta, \Delta) \end{array} \right) \\ & \wedge \left(\begin{array}{c} \hat{y}' = T' \wedge \mathrm{PeriodicUpdate}(T, \mathrm{ValMod}(\hat{y}, T, L_S), P, J, \phi) \\ \vee \\ \hat{y}' = \hat{y} \wedge \mathrm{PeriodicIdle}(T, \mathrm{ValMod}(\hat{y}, T, L_S), P, J, \phi) \end{array} \right) \end{aligned}$$

Définition 88. *Condition d'anticipation d'un chemin. Soit une spécification* $\langle \mathcal{A}\mathrm{rchi}, \mathcal{P}\mathrm{rop} \rangle$, L_S *la longueur de l'intervalle d'analyse de cette spécification et un chemin* $[x] : P : [z, y]$ *d'une variable* x *source vers une variable* y *et tel qu'il existe* Z *tel que* $z \in Z$ *et* $y \preccurlyeq f(Z)$*. On suppose que l'on a la propriété suivante sur le chemin* P*.*

$$P \left\{ \begin{array}{l} \mathrm{Latency}(\delta_{Lat}, \Delta_{Lat}), \mathrm{Lag}(\delta_{Lag}, \Delta_{Lag}), \mathrm{Shift}(\delta_{Shi}, \Delta_{Shi}), \\ \mathrm{Freshness}(\delta_{Fre}, \Delta_{Fre}), \mathrm{Fitness}(\delta_{Fit}, \Delta_{Fit}), \mathrm{Stability}(\delta_{Sta}, \Delta_{Sta}) \end{array} \right\}$$

L'action du chemin P *est définie par :*

$$\begin{aligned} & A_P \triangleq dur'_P \triangleq \\ & \left\{ \begin{array}{l} \langle \max\left(\begin{array}{c} \delta_{Sta}, \delta_{Fre} + \delta_{Fit} + 1, \\ T' - upd + \delta_{Fit} - \Delta_{Shi} + 2 \end{array} \right), \min\left(\begin{array}{c} \Delta_{Sta} - 1, \Delta_{Fre} + \Delta_{Fit} - 1, \\ T' - upd + \Delta_{Fit} - \delta_{Shi} \end{array} \right) \rangle \\ \textit{si } \langle \hat{x}', T' + 1 - \mathrm{ValMod}(\hat{x}', T', L_S), [x] : P : [z, y] \rangle \in \min(H'_{y \preccurlyeq f(Z)}) \wedge \\ \quad \langle \hat{x}, T + 1 - \mathrm{ValMod}(\hat{x}', T', L_S), [x] : P : [z, y] \rangle \notin \min(H_{y \preccurlyeq f(Z)}) \\ \langle \max\left(\begin{array}{c} \delta, \delta_{Sta}, \delta_{Fre} + \delta_{Fit} + 1, \\ T' - upd + \delta_{Fit} - \Delta_{Shi} + 2 \end{array} \right), \min\left(\begin{array}{c} \delta, \Delta_{Sta} - 1, \Delta_{Fre} + \Delta_{Fit} - 1, \\ T' - upd + \Delta_{Fit} - \delta_{Shi} \end{array} \right) \rangle \\ \textit{si } \langle \hat{x}', T' + 1 - \mathrm{ValMod}(\hat{x}', T', L_S), [x] : P : [z, y] \rangle \in \min(H'_{y \preccurlyeq f(Z)}) \wedge \\ \quad \langle \hat{x}, T + 1 - \mathrm{ValMod}(\hat{x}', T', L_S), [x] : P : [z, y] \rangle \in \min(H_{y \preccurlyeq f(Z)}) \wedge \\ \quad dur_P = \langle \delta, \Delta \rangle \\ \langle 0, +\infty \rangle \textit{ sinon} \end{array} \right. \end{aligned}$$

Définition 89. *Historique des occurrences Pour une spécification* $\langle \mathcal{A}\mathrm{rchi}, \mathcal{P}\mathrm{rop} \rangle$*,* L_S *la longueur de l'intervalle d'analyse de cette spécification et une exécution* σ *satisfaisant cette spécification, pour une variable* y*, image d'une observation* $y \preccurlyeq f(X)$*. L'action de l'historique d'une observation* $y \preccurlyeq f(X)$ *est définie par :*

$$\begin{aligned} & A_{H_{y \preccurlyeq f(X)}} \triangleq \\ & \quad \wedge \forall j \in [\mathrm{Index}(\min(H'_{y \preccurlyeq f(X)}))..i] : \\ & \qquad . \ H_{y \preccurlyeq f(X)}[j]' = \left\{ o \left| \begin{array}{l} \exists o' \in H_{y \preccurlyeq f(X)}[j] : o' = \langle upd, dur, [z] : P \rangle \wedge \\ \left(\begin{array}{l} (\hat{z} \neq upd \wedge o = \langle upd, dur, [z] : P \rangle) \vee \\ (\hat{z} = upd \wedge o = \langle upd, T' + 1 - \mathrm{ValMod}(\hat{z}', T, L_S), [z] : P \rangle) \end{array} \right) \end{array} \right. \right\} \\ & \quad \wedge H_{y \preccurlyeq f(X)}[\mathrm{IndexMax}(H_{y \preccurlyeq f(X)}) + 1]' = \{ \langle \hat{x}', T' + 1 - \mathrm{ValMod}(\hat{x}', T', L_S), [x, y] \rangle | x \in X \} \cup \\ & \quad \left\{ o \left| \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in \min(H'_{x \preccurlyeq g(Z)}) \\ \wedge\ o' = \langle upd, dur, P \rangle \wedge o = \langle upd, dur, P : [y] \rangle \wedge P : [y] \in \mathcal{P}aths(\mathcal{A}rchi) \end{array} \right. \right\} \\ & \wedge \left(\begin{array}{c} \hat{y}' = T' \wedge \min(H'_{y \preccurlyeq f(X)}) \neq \min(H'_{y \preccurlyeq f(X)}) \\ \vee \\ \hat{y}' = \hat{y} \wedge \min(H'_{y \preccurlyeq f(X)}) = \min(H'_{y \preccurlyeq f(X)})) \end{array} \right) \end{aligned}$$

Définition 90. *Action d'une image sporadique. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification, tel qu'il existe une observation* $y \preccurlyeq f(X)$ *dans l'architecture du système. L'action de la variable* y *est définie par :*

$$
\begin{array}{l}
A_y \triangleq \\
\quad \wedge A_{H_{y \preccurlyeq f(X)}} \ \wedge A_{dur_y} \\
\quad \wedge \textit{Let } \langle \delta, \Delta \rangle = dur'_y \textit{ in} \\
\qquad \vee \hat{y}' = T' \ \wedge \ SporadicUpdate(T, ValMod(\hat{y}, T, L_S), min(\delta, \delta_{Spo}), max(\Delta, \Delta_{Spo})) \\
\qquad \vee \hat{y}' = \hat{y} \ \wedge \ SporadicIdle(T, ValMod(\hat{y}, T, L_S), min(\delta, \delta_{Spo}), max(\Delta, \Delta_{Spo})) \\
\quad \wedge \forall \langle upd, dur, [z] : P \rangle \in min(H'_{y \preccurlyeq f(X)}) : \\
\qquad \wedge A_{[z]:P} \\
\qquad \wedge \ \ \vee (upd = 0 \ \wedge \ T' < \Delta_{[z]:P}) \\
\qquad\quad \vee \ \ \wedge upd \neq 0 \\
\qquad\qquad \wedge (upd \neq \hat{z}' \Rightarrow dur \in DurationInterval(T', VM, PP) \\
\qquad\qquad \wedge \ \ \vee (\hat{y}' = T' \ \wedge \ T' \in \left(\begin{array}{c} PathInterval(VM, PP) \cap \\ [VM + MinLag([z] : P)..VM + MaxLag([z] : P)[\end{array} \right) \\
\qquad\qquad\quad \vee \hat{y}' = \hat{y} \ \wedge \ T' \in PathInterval(VM, PP)
\end{array}
$$

où l'on a : $VM = ValMod(upd, T', L_S)$ *et* $PP = PathParameters([z] : P)$

Définition 91. *Action d'une image périodique. Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification, tel qu'il existe une observation* $y \preccurlyeq f(X)$ *dans l'architecture du système et tel qu'il existe une propriété de périodicité sur cette variable :*

$$y \ \{Periodic(P, J, \phi)\} \in \mathcal{P}rop$$

L'action de la variable y *est définie par :*

$$
\begin{array}{l}
A_y \triangleq \\
\quad \wedge A_{H_{y \preccurlyeq f(X)}} \wedge A_{dur_y} \\
\quad \wedge \textit{Let } \langle \delta, \Delta \rangle = dur'_y \textit{ in} \\
\qquad \vee \hat{y}' = T' \ \wedge \ SporadicUpdate(T, ValMod(\hat{y}, T, L_S), \delta, \Delta) \\
\qquad \vee \hat{y}' = \hat{y} \ \wedge \ SporadicIdle(T, ValMod(\hat{y}, T, L_S), \delta, \Delta) \\
\quad \wedge \ \ \vee \hat{y}' = T' \ \wedge \ PeriodicUpdate(T, ValMod(\hat{y}, T, L_S), P, J, \phi) \\
\qquad \vee \hat{y}' = \hat{y} \ \wedge \ PeriodicIdle(T, ValMod(\hat{y}, T, L_S), P, J, \phi) \\
\quad \wedge \forall \langle upd, dur, [z] : P \rangle \in min(H'_{y \preccurlyeq f(X)}) : \\
\qquad \wedge A_{[z]:P} \wedge \\
\qquad \wedge \ \ \vee upd = 0 \ \wedge \ T' < \Delta_{[z]:P} \\
\qquad\quad \vee \ \ \wedge upd \neq 0 \\
\qquad\qquad \wedge (upd \neq \hat{z}' \Rightarrow dur \in DurationInterval(T', VM, PP) \\
\qquad\qquad \wedge \ \ \vee \hat{y}' = T' \ \wedge \ T' \in \left(\begin{array}{c} PathInterval(VM, PP) \cap \\ [VM + MinLag([z] : P)..VM + MaxLag([z] : P)[\end{array} \right) \\
\qquad\qquad\quad \vee \hat{y}' = \hat{y} \ \wedge \ T' \in PathInterval(VM, PP)
\end{array}
$$

où l'on a : $VM = ValMod(upd, T', L_S)$ *et* $PP = PathParameters([z] : P)$

7.2.3 *Propriétés*

On donne ici un ensemble de propriétés utilisées pour prouver par la suite l'équivalence entre le système de transitions fini et le système de transitions défini au chapitre 6. On veut notamment prouver que dans les exécutions du système de transitions réduit,

les différences définissant l'action des variables de l'architecture ne dépassent pas la longueur la longueur de l'intervalle d'analyse.

Lemme 47. *Si on a* $-L_S < T - upd < L_S$*, alors :*

$$ValMod(upd, T, L_S) = \begin{cases} upd \textit{ si } upd \leq T \\ upd - L_S \textit{ sinon} \end{cases}$$

Démonstration. On suppose que $-L_S < T - upd < L_S$. Si on a $upd \leq T$, alors :

$$\begin{array}{ll} & 0 \leq T - upd < L_S \\ \Rightarrow & \{\text{propriété du modulo}\} \\ & ((T - upd) \pmod{L_S}) = T - upd \\ \Rightarrow & \{\text{multiplication par } -1 \text{ et ajout de T}\} \\ & -((T - upd) \pmod{L_S}) + T = upd \\ \Rightarrow & \{\text{définition de } ValMod\} \\ & ValMod(upd, T, L_S) = upd \end{array}$$

Si on a $T < upd$, alors :

$$\begin{array}{ll} & -L_S < T - upd < 0 \\ \Rightarrow & \{\text{propriété du modulo}\} \\ & ((T - upd) \pmod{L_S}) = T - upd + L_S \\ \Rightarrow & \{\text{multiplication par } -1 \text{ et ajout de T}\} \\ & -((T - upd) \pmod{L_S}) + T = upd - L_S \\ \Rightarrow & \{\text{définition de } ValMod\} \\ & ValMod(upd, T, L_S) = upd - L_S \end{array}$$

□

On en déduit la proposition suivante :

Lemme 48. *Si on a* $-L_S < T - upd < L_S$ *et* $T \geq 0$*, alors :*

$$T - ValMod(upd, T, L_S) = ((T \pmod{L_S}) - upd \pmod{L_S}) \pmod{L_S}$$

Démonstration. On distingue deux cas selon le signe de $T - upd$. On suppose tout d'abord que l'on a $T - upd \geq 0$. Alors d'après le lemme 47, on a :

$$\begin{array}{ll} & T - ValMod(upd, T, L_S) = T - upd \\ \Rightarrow & \{\text{proposition 43}\} \\ & T - ValMod(upd, T, L_S) = (T \pmod{L_S}) - upd \pmod{L_S}) \pmod{L_S} \end{array}$$

Si jamais on a $T - upd < 0$, alors on a :

$$\begin{array}{ll} & T - ValMod(upd, T, L_S) = T - upd + L_S \\ \Rightarrow & \{\text{lemme 43}\} \\ & T - ValMod(upd, T, L_S) = ((T + L_S \pmod{L_S}) - upd \pmod{L_S}) \pmod{L_S} \\ \Rightarrow & \{(T + L_S) \pmod{L_S} = T \text{ car T et } L_S \text{ positif}\} \\ & T - ValMod(upd, T, L_S) = (T \pmod{L_S}) - upd \pmod{L_S}) \pmod{L_S} \end{array}$$

□

On montre désormais que les différences utilisées pour définir les actions associées à chaque variable vérifient en chaque état les conditions d'application du lemme 43.

Lemme 49. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$, L_S la longueur de l'intervalle d'analyse de cette spécification et $(\Sigma, \Sigma_0, \rightarrow_{L_S})$ le système de transitions réduit défini par cette spécification. Pour toute exécution σ de ce système de transition, on a :*

$$\begin{array}{l}\forall i \in \mathbb{N} : \\ \forall obs \in \mathcal{A}, \forall \langle upd, dur, P\rangle \in H_{obs}.\sigma_i : -L_S < T.\sigma_i - upd < L_S - 1 \wedge \\ \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : -L_S < T.\sigma_i - \hat{\hat{x}}.\sigma_i < L_S - 1\end{array}$$

Pour prouver ce lemme, on prouve que l'on a aussi :

Lemme 50. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop\rangle$, L_S la longueur de l'intervalle d'analyse de cette spécification et $(\Sigma, \Sigma_0, \rightarrow_{L_S})$ le système de transitions réduit défini par cette spécification. Pour toute exécution σ de ce système de transition, on a :*

$$\begin{array}{l}\forall i \in \mathbb{N} : \\ \forall obs \in \mathcal{A}, \forall \langle upd, dur, P\rangle \in H_{obs}.\sigma_i : 0 \leq T.\sigma_i - ValMod(upd, T.\sigma_i, L_S) < L_S - 1 \wedge \\ \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : 0 \leq T.\sigma_i - ValMod(\hat{\hat{x}}.\sigma_i, T.\sigma_i, L_S) < L_S - 1\end{array}$$

Démonstration. On démontre ces deux lemmes par récurrence. On prouve de plus par récurrence que pour tout i, s'il n'y a pas de mise à jour entre σ_i et σ_{i+1}, alors :

$$T.\sigma_{i+1} - ValMod(\hat{\hat{x}}.\sigma_{i+1}, T.\sigma_{i+1}, L_S) = T.\sigma_i - ValMod(\hat{\hat{x}}.\sigma_i, T.\sigma_i, L_S) + 1$$

Dans l'état initial, on a :

$$\begin{array}{l}\forall obs \in \mathcal{A}rchi, \forall \langle upd, dur, P\rangle \in H_{obs}.\sigma_i : T.\sigma_0 - upd = 0 \wedge \\ \forall x \in \mathcal{V}ar(\mathcal{A}rchi) : T.\sigma_0 - \hat{\hat{x}}.\sigma_0 = 0\end{array}$$

D'après le lemme 48, les différences entre T et la fonction $ValMod$ appliqué à upd ou à $\hat{\hat{x}}$ sont aussi égales à 0.

De plus, s'il n'y a pas de mise à jour de x, alors on a :

$$T.\sigma_1 - ValMod(\hat{\hat{x}}.\sigma_1, T.\sigma_1, L_S) = 1 - ValMod(0, 1, L_S) = 1 = T.\sigma_0 - ValMod(\hat{\hat{x}}.\sigma_0, T.\sigma_0, L_S) + 1$$

La différence est bien incrémentée de 1.

Soit un entier i. On suppose désormais que cette propriété ainsi que les lemmes sont vraies dans tous les états antérieurs i. On prouve la propriété en s'intéressant d'abord à la différence entre T et le profil temporel d'une variable puis en s'intéressant à la différence entre T et la date de mise à jour d'une occurrence dans l'historique d'une observation.

1. On considère tout d'abord une variable x de l'architecture. Il y alors deux possibilités, soit x et donc la valeur de $\hat{\hat{x}}$ est mise à jour entre σ_i et σ_{i-1}, soit elle ne l'est pas. Si elle est mise à jour, $\hat{\hat{x}}$ prend la valeur de T et on a donc :

$$T.\sigma_i - \hat{\hat{x}}.\sigma_i = 0 = T.\sigma_i - ValMod(\hat{\hat{x}}.\sigma_i, T.\sigma_i, L_S)$$

Donc la propriété est vérifiée.

Si jamais $\hat{\hat{x}}$ n'est pas mis à jour, on a $\hat{\hat{x}}.\sigma_i = \hat{\hat{x}}.\sigma_{i-1}$. Par hypothèse de récurrence, on a :

$$-L_S < T.\sigma_{i-1} - \hat{\hat{x}}.\sigma_{i-1} < L_S - 1$$

On prouve alors que cette propriété est aussi vraie en σ_i alors que $\hat{\hat{x}}$ n'est pas mis à jour.

1.A. On prouve tout d'abord que lors du passage de σ_{i-1} à σ_i, soit T est incrémenté de 1, soit il est décrémenté de $L_S - 1$ mais dans les deux cas, on a :

$$T.\sigma_i - ValMod(\hat{x}.\sigma_i, T.\sigma_i, L_S) = T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S) + 1$$

Il y a plusieurs cas selon le signe de $T.\sigma_{i-1} - \hat{x}.\sigma_{i-1}$.

1.A.1. Premier cas si on a :

$$0 \leq T.\sigma_{i-1} - \hat{x}.\sigma_{i-1} < L_S - 1$$

D'après le lemme 47, on a alors :

$$T.\sigma_{i-1} - \hat{x}.\sigma_{i-1} = T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S)$$

1.A.1.A. Si T est incrémenté de 1, on a alors :

$$1 \leq T.\sigma_i - \hat{x}.\sigma_i < L_S$$

Alors d'après le lemme 47 :

$$T.\sigma_i - \hat{x}.\sigma_i = T.\sigma_i - ValMod(\hat{x}.\sigma_i, T.\sigma_i, L_S) = T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S) + 1$$

1.A.1.B. Si T est décrémenté de $L_S - 1$ c'est que $T.\sigma_{i-1} + 1 = 2 * L_S - 1$ et on a alors :

$$-L_S + 1 \leq T.\sigma_i - \hat{x}.\sigma_i < 0$$

Et donc d'après le lemme 47 :

$$T.\sigma_i - ValMod(\hat{x}.\sigma_i, T.\sigma_i, L_S) = T.\sigma_i - \hat{x}.\sigma_i + L_S = 2 * L_S - \hat{x}.\sigma_i$$

Comme $T.\sigma_{i-1} = 2 * L_S - 1$ et $\hat{x}.\sigma_i = \hat{x}.\sigma_{i-1}$, on a bien :

$$T.\sigma_i - ValMod(\hat{x}.\sigma_i, T.\sigma_i, L_S) = T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S) + 1$$

1.A.1. On suppose désormais que :

$$-L_S + 1 < T.\sigma_{i-1} - \hat{x}.\sigma_{i-1} < 0$$

On a alors :

$$T.\sigma_{i-1} - \hat{x}.\sigma_{i-1} + L_S = T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S)$$

On a $T.\sigma_{i-1} < \hat{x}.\sigma_{i-1}$. Comme la valeur de $\hat{x}$ a été fixée par une valeur de T dans un état précédent et que T est strictement inférieur à $2 * L_S$, on en déduit que $T.\sigma_{i-1} < 2 * L_S - 1$. Donc nécessairement lors du passage de σ_{i-1} à σ_i, T est incrémenté de 1. On a donc :

$$-L_S + 2 < T.\sigma_i - \hat{x}.\sigma_i < 1$$

Si cette différence est strictement négative, on a :

$$T.\sigma_i - ValMod(\hat{x}.\sigma_i, T.\sigma_i, L_S) = T.\sigma_i - \hat{x}.\sigma_i + L_S = T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S) + 1$$

Si elle est égale à 0, alors c'est qu'on a $T.\sigma_i = \hat{x}.\sigma_i$ et donc $T.\sigma_{i-1} + 1 = \hat{x}.\sigma_{i-1}$ et donc :

$$T.\sigma_{i-1} - ValMod(\hat{x}.\sigma_{i-1}, T.\sigma_{i-1}, L_S) = L_S - 1$$

On montre que ce cas n'est pas possible. Il y a deux possibilités :

– si pour x il y a une propriété de sporadicité avec une borne supérieure finie sur la sporadicité, cette borne est strictement inférieure à $L_S - 1$. Lors des transitions entre les états antérieurs, lorsque x n'était pas mis à jour, la différence est incrémentée. Il y a donc eu un état antérieur où la différence entre T et $ValMod(\hat{x}, T, L_S$ est égale à la borne supérieure sur la sporadicité de x. Cette différence en atteignant la borne supérieure doit alors entraîner une mise à jour de $\hat{x}$ d'après la définition de l'action de x. La différence ne peut alors croître au delà de cette borne et ne peut être égale à $L_S - 1$
– si il n'y a pas de telle propriété de sporadicité pour x, c'est, d'après les conditions pour que le temps de sortie du régime initial soit borné, que x est image d'une observation. Il y a alors une propriété de décalage ou de latence avec une borne supérieure bornée sur un chemin dont x est la source ou une propriété de latence sur un chemin passant par x. La valeur de $\hat{x}$ est propagée le long d'un tel chemin. Tant que x n'est pas mis à jour, il n'y pas d'occurrence à la source d'un tel chemin plus récente que celle sur laquelle est basée l'occurrence de x apparue en $\hat{x}.\sigma_i$ et disponible pour définir l'occurrence de l'image du chemin. Or lorsque les bornes supérieures des propriétés de ce chemin sont atteinte, cette occurrence n'est plus assez fraîche pour satisfaire ces propriétés. $\hat{x}$ doit donc nécessairement être mis à jour avant que les bornes de ces propriétés soient atteintes, bornes strictement inférieur à $L_S - 1$.

Ce cas n'est donc pas possible. Donc dans tous les cas cette différence est incrémentée.

1.B On s'intéresse de nouveau à la différence $T.\sigma_i - \hat{x}.\sigma_{i-1}$. On suppose que l'on a par hypothèse de récurrence :

$$-L_S < T.\sigma_{i-1} - \hat{x}.\sigma_{i-1} < L_S - 1$$

et on veut prouver que l'on a :

$$-L_S < T.\sigma_i - \hat{x}.\sigma_i < L_S - 1$$

alors que $\hat{x}$ n'est pas mis à jour. On considère ici deux cas selon l'évolution de T.

1.B.1. Si T est incrémenté de 1 entre σ_{i-1} et σ_i, on a :

$$-L_S + 1 < T.\sigma_i - \hat{x}.\sigma_i < L_S$$

Supposons qu'on a : $T.\sigma_{i-1} - \hat{x}.\sigma_{i-1} = L_S - 1$. La différence $T.\sigma_i - ValMod(\hat{x}.\sigma_i, T.\sigma_i, L_S)$ est alors égale à L_S. Or, lors des transitions précédentes, elle a été incrémentée de 1 à chaque fois. Il y a donc eu une transition où la différence était égale à la borne maximum de la sporadicité de x et donc $\hat{x}$ devait être mis à jour et la différence devait repasser à 0. Ce cas n'est donc pas possible et on a donc bien :

$$-L_S + 1 < T.\sigma_i - \hat{x}.\sigma_i < L_S - 1$$

1.B.2. Si T est décrémenté de $L_S - 1$ entre σ_{i-1} et σ_i, c'est qu'on a :

$$T.\sigma_{i-1} + 1 = 2 * L_S$$

Comme les valeurs de $\hat{\hat{x}}$ sont fixée par les valeur de T et donc ne peuvent être supérieure ou égal à $2 * L_S$, on a alors nécessairement :

$$
\begin{array}{ll}
 & T.\sigma_{i-1} + 1 > \hat{\hat{x}}.\sigma_{i-1} \\
\Rightarrow & \quad \{\text{réécriture}\} \\
 & T.\sigma_{i-1} - \hat{\hat{x}}.\sigma_{i-1} \geq 0 \\
\Rightarrow & \quad \{\text{hypothèse de récurrence : } -L_S + 1 < T.\sigma_{i-1} - \hat{\hat{x}}.\sigma_{i-1} < L_S - 1\} \\
 & 0 \leq T.\sigma_{i-1} - \hat{\hat{x}}.\sigma_{i-1} < L_S - 1
\end{array}
$$

Donc si T est décrémenté de $L_S - 1$, on a :

$$-L_S + 1 \leq T.\sigma_i - \hat{\hat{x}}.\sigma_{i-1} < 0$$

La propriété est alors bien vérifiée. Donc dans tous les cas, on a bien :

$$-L_S < T.\sigma_i - \hat{\hat{x}}.\sigma_i < L_S - 1.$$

2. On s'intéresse aux date de mise à jours dans les caractéristiques des occurrences dans l'historique d'une observation. Soit une observation $y \lessdot f(X)$ et soit $\langle upd, dur, P \rangle$ un triplet appartenant à $H_{y \lessdot f(X)}.\sigma_i$.

Pour effectuer la preuve des propositions concernant la différence entre T et $ValMod(\hat{\hat{x}}.\sigma_j, T.\sigma_j, L_S)$, on a prouvé qu'entre deux états σ_j et σ_{j+1}, cette différence était incrémentée de 1 jusqu'à atteindre la borne supérieure de la sporadicité de x. On prouve que pour le triplet $\langle upd, dur, P \rangle$ la différence entre T et $ValMod(upd, T.\sigma_j, L_S)$ est aussi incrémentée de 1. Pour cela, on a une preuve similaire à cette entre T et $\hat{\hat{x}}$. Cependant, la différence, au lieu d'être bornée par la sporadicité, est ici bornée par la latence du chemin P ou la latence d'un sur-chemin de P. On prouve l'existence d'une telle borne.

Les triplets $\langle upd', dur', P \rangle$ qui concernent un chemin P sont à des indices différents de $H_{y \lessdot f(X)}.\sigma_i$. Les triplets stockés à des indices supérieures ont été ajoutés plus tard et concernent des occurrences de la source de P apparues plus tard. Donc la différence $T.\sigma_i - ValMod(upd', T.\sigma_i, L_S)$ est inférieur à celle défini par upd. Inversement, les triplets à des indices supérieurs correspondent à des occurrences de la source de P apparues plus tôt et donc la différence $T.\sigma_i - ValMod(upd', T.\sigma_i, L_S)$ est supérieure. Le triplet à l'indice minimum a donc une différence $T.\sigma_i - ValMod(upd', T.\sigma_i, L_S)$ supérieure aux autres triplets correspondant au même chemin dans l'historique $H_{y \lessdot f(X)}$. On borne la différence pour ce triplet à l'indice minimum. On a alors deux cas :

- On suppose qu'il y a une propriété de latence spécifiée sur ce chemin. L'action A_y spécifie que pour le triplet $\langle upd', dur', P \rangle$ donnant les caractéristiques de P et qui est stockée à l'indice minimum de $H_{y \lessdot f(X)}.\sigma_i$, la différence $T.\sigma_i - ValMod(upd', T.\sigma_i, L_S)$ doit vérifier la propriété de latence. En effet T doit appartenir à l'intervalle PathInterval. La différence est donc bornée par cette latence strictement inférieure à L_S.
- On suppose qu'il y a une propriété de latence spécifiée sur un sur-chemin de P. Soit P′ ce chemin. Il existe une observation obs définissant le dernier arc de ce chemin. Dans l'historique de cette observation, le triplet $\langle upd'', dur'', P' \rangle$ caractérisant le chemin P′ et stockée au minimum de l'historique est tel que la différence $T.\sigma_i - ValMod(upd'', T.\sigma_i, L_S)$ est bornée par la latence de P′′. Or ce triplet correspond à une occurrence apparue avant upd′. En effet, il n'est pas possible qu'une occurrence ait fini de se propager le long de P′ avant de s'être propagée le long de P, P′ étant un sur-chemin de P. La différence $T.\sigma_i - ValMod(upd', T.\sigma_i, L_S)$ est donc bornée par celle définie par upd′′ et donc est strictement inférieure à L_S.

Alors, en utilisant la même preuve que pour la différence entre T et $\hat{x}$ pour une variable sporadique et pour laquelle la différence $T - ValMod(\hat{x}, T, L_S)$ est bornée par la borne supérieure de la sporadicité, on peut prouve qu'ici aussi, on a $-L_S < T.\sigma_i - upd < L_S - 1$. □

On déduit alors des propositions précédentes, que l'on a :

Proposition 51. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, L_S la longueur de l'intervalle d'analyse de cette spécification et $(\Sigma, \Sigma_0, \rightarrow_{L_S})$ le système de transitions réduit défini par cette spécification. Pour toute exécution σ de ce système de transition, on a :*

$$\begin{array}{l}
\forall \sigma_i, \forall upd \in \mathbb{N} : \\
\left(\begin{array}{c} (\exists obs \in \mathcal{A} : \exists \langle upd, dur, P \rangle \in H_{obs}.\sigma_i \ \vee \ \exists x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{x}.\sigma_i = upd) \wedge \\ (\exists T', upd' \in \mathbb{N} : \left(\begin{array}{c} T' \pmod{L_S} = T.\sigma_i \pmod{L_S} \wedge \\ upd' \pmod{L_S} = upd \pmod{L_S} \ \wedge \ T' - upd' \in [0..L_S[\end{array} \right) \end{array} \right) \\
\Rightarrow T' - upd' = T.\sigma_i - ValMod(upd, T.\sigma_i, L_S)
\end{array}$$

Démonstration. La preuve est déduite des lemmes 49, 48 et 43. □

Proposition 52. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, L_S la longueur de l'intervalle d'analyse de cette spécification et $(\Sigma, \Sigma_0, \rightarrow_{L_S})$ le système de transitions réduit défini par cette spécification. Pour toute exécution σ de ce système de transition, on a :*

$$\begin{array}{l}
\forall i \in \mathbb{N}, \forall upd_1, upd_2 : \\
(\exists obs \in \mathcal{A} : \exists \langle upd_1, dur, P \rangle \in H_{obs}.\sigma_i \ \vee \ \exists x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{x}.\sigma_i = upd_1) \wedge \\
(\exists obs \in \mathcal{A} : \exists \langle upd_2, dur, P \rangle \in H_{obs}.\sigma_i \ \vee \ \exists x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{x}.\sigma_i = upd_2) \wedge \\
upd_1 \pmod{L} = upd_2 \pmod{L} \Rightarrow upd_1 = upd_2
\end{array}$$

Démonstration. Soit une exécution de $(\Sigma, \Sigma_0, \rightarrow_{L_S})$. On effectue une preuve par l'absurde, on suppose donc que l'on a pour un état σ_i deux entiers upd_1 et upd_2 tel que :

$$\begin{array}{l}
(\exists obs \in \mathcal{A} : \exists \langle upd_1, dur, P \rangle \in H_{obs}.\sigma_i \ \vee \ \exists x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{x}.\sigma_i = upd_1) \wedge \\
(\exists obs \in \mathcal{A} : \exists \langle upd_2, dur, P \rangle \in H_{obs}.\sigma_i \ \vee \ \exists x \in \mathcal{V}ar(\mathcal{A}rchi) : \hat{x}.\sigma_i = upd_2) \wedge \\
upd_1 \pmod{L} = upd_2 \pmod{L} \ \wedge \ upd_1 \neq upd_2
\end{array}$$

On suppose ici que $upd_1 < upd_2$. On sait d'après le lemme 49 que l'on a :

$$-L_S + 1 < T.\sigma_i - upd_1 < L_S - 1 \ \wedge \ -L_S + 1 < T.\sigma_i - upd_2 < L_S - 1$$

Comme les valeurs de ces deux entiers sont fixées par des valeurs prises par T, on sait de plus que $upd_1, upd_2 \in [0..2 * L_S[$. Comme ces deux valeurs sont égales modulo L_S, on en déduit donc que $upd_2 = upd_1 + L_S$. On considère trois cas :

Les valeurs de ces deux entiers ont étés fixés par copie de la valeur de T. On suppose tout d'abord que upd_1 est la première valeur à être apparue dans le système. Lorsque upd_2 est apparue, on avait un état σ_j tel que $T.\sigma_j = upd_2 = upd_1 + L_S$. On n'avait alors pas $T.\sigma_i - upd_1 < L_S - 1$ et donc ce n'est pas possible.

Si jamais c'est upd_2 la première valeur apparue dans le système, alors lorsque upd_1 est apparue, on avait un état σ_j tel que $T.\sigma_j = upd_1 = upd_2 - L_S$. On n'avait alors pas $T.\sigma_i - upd_2 > -L_S + 1$ et donc ce n'est pas possible.

On a donc nécessairement $upd_2 = upd_1$. □

On donne finalement une dernière proposition sur le régime initial des variables et des chemins du système de transitions réduit.

Proposition 53. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, L_S *la longueur de l'intervalle d'analyse de cette spécification et* $(\Sigma, \Sigma_0, \rightarrow_{L_S})$ *le système de transitions réduit défini par cette spécification. Pour toute exécution* σ *de ce système de transition, on a :*

$$\forall x \in \mathcal{V}ar(\mathcal{A}rchi), \forall i \in \mathbb{N} : \hat{\hat{x}}.\sigma_i = 0 \Rightarrow T.\sigma_i < L_S$$

De même, pour toute observation $y \lesssim f(X)$ *on a :*

$$\forall i \in \mathbb{N}, \forall \langle upd, dur, P \rangle \in H_{y \lesssim f(X)}.\sigma_i : upd = 0 \Rightarrow T.\sigma_i < L_S$$

Démonstration. La relation de transition $\rightarrow_{L_S}$ est défini tel que T croît à chaque transition jusqu'à la valeur $2 * L_S$ où elle repasse à L_S et reste ensuite dans l'intervalle $[L_S..2 * L_S[$. Le seul état σ_i où l'on a $T.\sigma_i = 0$, c'est donc l'état initial. Les valeurs de $\hat{\hat{x}}$ sont celles de T lors des mises à jour. $\hat{\hat{x}}$ ne peut donc pas prendre la valeur 0 lors d'une mise à jour. Donc si dans un état σ_i, on a $\hat{\hat{x}}.\sigma_i = 0$ c'est qu'il n'y pas eu de mise à jour de x depuis l'état initial. Or les actions des différentes variables forcent une mise à jour avant que le temps de sortie du régime initial de ces variables soit atteint. Pour un état σ_i, si on a $\hat{\hat{x}}.\sigma_i = 0$, alors nécessairement $T.\sigma_i < \Delta_x \leq L_S$.

De même, pour une variable historique d'une observation, lorsque une caractéristique avec une date de mise à jour upd nulle est placée dans cette historique c'est que la source est dans le régime initial. Les occurrences ajoutées par la suite ne sont plus dans le régime initial. Seule les premières caractéristiques ajoutées peuvent donc avoir une date de mise à jour upd nul. Ces caractéristiques sont nécessairement stockées à l'indice minimum du tableau historique, les indices les plus faibles correspondant aux caractéristiques ajoutées en premier. Or ces caractéristiques doivent vérifier les propriétés du chemin et donc si $upd = 0$, on a donc nécessairement :

$$T.\sigma_i < L_S$$

□

7.2.4 *Équivalence des systèmes de transitions*

Proposition 54. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification. Soit* $(\Sigma, \Sigma_0, \rightarrow)$ *le système de transitions défini par cette spécification et* $(\Sigma, \Sigma_0, \rightarrow_{L_S})$ *le système de transitions du système réduit. On a :*

$$\begin{array}{c} \forall \sigma \in [\![\langle (\Sigma, \Sigma_0, \rightarrow) \rangle]\!]_{ST}, \exists \sigma' \in [\![(\Sigma, \Sigma_0, \rightarrow_{L_S})]\!]_{ST} : \forall i \in \mathbb{N} : \sigma_i \sim_{L_S} \sigma'_i \\ \wedge \\ \forall \sigma' \in [\![(\Sigma, \Sigma_0, \rightarrow_{L_S})]\!]_{ST}, \exists \sigma \in [\![(\Sigma, \Sigma_0, \rightarrow)]\!]_{ST} : \forall i \in \mathbb{N} : \sigma_i \sim_{L_S} \sigma'_i \end{array}$$

Les sémantiques des deux systèmes de transitions sont donc équivalentes. Pour prouver cette équivalence, on prouve l'équivalence des exécutions car c'est uniquement dans le cadre de ces exécutions et des états de ces exécutions que les différences entre les différentes variables sont bornées par la longueur de l'intervalle d'analyse.

Démonstration. Pour démontrer cette propriété, on procède en deux étapes démontrant les deux parties de la proposition. Il s'agit dans les deux cas de considérer une exécution d'un des systèmes et de montrer que l'on peut construire une exécution équivalente. Construire une telle exécution est possible lorsque l'on peut construire un état tel que dans les deux exécutions ce sont les mêmes prédicats qui définissent les actions de chaque variable qui sont vérifiées lors des transitions de chaque système.

1. On démontre par récurrence sur les états que pour toute exécution définie par le système de transitions équivalent à une spécification, il existe une exécution équivalente du système de transitions réduit. On suppose que l'on a une exécution σ définie par le système de transitions équivalent à une spécification. On prouve que si pour un entier n, on peut pour tout k inférieur ou égal à n construire un état σ'_k équivalent à l'état σ_k et tel que $\sigma'_{k-1} \rightarrow_{L_S} \sigma'_k$, alors on pourra construire un état σ'_{n+1} équivalent à σ_{n+1} et tel que $\sigma'_n \rightarrow_{L_S} \sigma'_{n+1}$. Pour cela, il faut prouver l'équivalence des actions définissant le comportement de chaque variable du système.

1.A. On choisit pour l'exécution σ' le même état initial que l'exécution σ. Ces deux états sont équivalents car l'ensemble des valeurs des variables sont égales et donc égales modulo L_S.

1.B. Soit un entier n. On suppose que pour tout k inférieur ou égal à n, on a construit un état σ'_k équivalent à l'état σ_k et tel que $\sigma'_{k-1} \rightarrow_{L_S} \sigma'_k$. On prouve que chacune des actions du système de transitions définie par la spécification qui est vérifiée entre σ_n et σ_{n+1}, est aussi vérifiée en effectuant les mêmes modifications entre σ'_n et σ'_{n+1}, c'est-à-dire en mettant les mêmes variables à jour avec des occurrences ayant des caractéristiques similaires.

On s'intéresse tout d'abord à la variable T. Entre σ_n et σ_{n+1}, elle est incrémentée de 1. Pour un état σ'_n et un état σ'_{n+1} tel que $\sigma'_n \rightarrow_{L_S} \sigma'_{n+1}$, T est aussi incrémentée de 1 sauf dans le cas où $T.\sigma'_n + 1 = 2 * L_S$. Dans ce cas on a la valeur de T qui est décrémentée de $L_S - 1$. Cependant, dans les deux cas, si les valeurs de T dans les états σ_n et σ'_n sont égales modulo L_S, alors elle le sont aussi dans les états σ_{n+1} et σ'_{n+1}.

Concernant les régimes initiaux, les propriétés d'une variable sont vérifiées car dans l'état σ_i on est dans le régime initial si on a :

$$\hat{x}.\sigma_n = 0 \wedge T.\sigma_n < \Delta_x$$

Or d'après la proposition 53, dans l'exécution σ' du système réduit, tant que $\hat{x}.\sigma'_n = 0$, on a $T.\sigma'_n < L_S$. On en déduit que les valeurs $T.\sigma_n$ et $T.\sigma'_n$ sont égales modulo L_S et inférieur à L_S et donc égales. Donc on a aussi $T.\sigma'_n < \Delta_x$. Dans les deux cas, les inégalités correspondant au régime initial sont donc respectées.

Pour les régimes initiaux des chemins, on a le même raisonnement. Pour les exécutions du système réduit, tant que la date de mise à jour dans une des caractéristique des occurrences d'un historique est nulle, c'est qu'on est dans un état $T.\sigma'_n$ tel que $T.\sigma'_n < L_S$. Donc cette valeur est égal à la valeur de T dans l'état équivalent σ_n. Dans les deux cas, les inégalités correspondant au régime initial des chemins sont donc respectées.

On continue cette preuve variable du système par variable et plus précisément, type de variable par type de variable. Soit une variable x du système, on différencie donc plusieurs cas selon les propriétés de x et la définition de l'action associée à cette variable. Pour chacune des propriétés, on ne s'intéresse pas au régime initial.

1.B.1. On suppose tout d'abord que la variable x est uniquement sporadique. L'évolution possible de x est définie en fonction des prédicats $\mathtt{SporadicUpdate}$ et $\mathtt{SporadicIdle}$. On montre que si entre σ_n et σ_{n+1} un de ces deux prédicats est vérifié, il est aussi possible de le vérifier entre σ'_n et un état σ'_{n+1} entre lesquels il y a une mise à jour si et uniquement s'il y en a une entre σ_n et σ_{n+1}.

Si on a $\hat{x}.\sigma'_n \neq 0$ et qu'on est donc en dehors du régime initial, on s'intéresse aux valeurs de $T.\sigma_n + 1 - \hat{x}.\sigma_n$ et $T.\sigma'_n + 1 - \mathtt{ValMod}(\hat{x}.\sigma'_n, T.\sigma'_n, L_S)$. D'après la proposition

51 ces deux différences sont égales, les valeurs de $T.\sigma_n$ et $T.\sigma'_n$ étant égales modulo L_S, les valeurs de $\hat{\hat{x}}.\sigma_n$ et $\hat{\hat{x}}.\sigma'_n$ étant aussi égales modulo L_S et la différence $T.\sigma_n + 1 - \hat{\hat{x}}.\sigma_n$ étant borné par L_S par définition de la valeur L_S choisi. On a donc :

$$\begin{array}{ll} & T.\sigma_n - \hat{\hat{x}}.\sigma_n = T.\sigma'_n - ValMod(\hat{\hat{x}}.\sigma'_n, T.\sigma'_n, L_S) \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & T.\sigma_n + 1 - \hat{\hat{x}}.\sigma_n = T.\sigma'_n + 1 - ValMod(\hat{\hat{x}}.\sigma'_n, T.\sigma'_n, L_S) \end{array}$$

Ces deux différences sont utilisées pour définir les prédicats $SporadicUpdate$ et $SporadicIdle$. Comme elles sont égales, si en σ_n l'un des prédicats est vérifié, il l'est aussi entre σ'_n. Si entre σ_n et σ_{n+1}, la variable x est mise à jour, on peut construire un états σ'_{n+1} en mettant aussi à jour x. La valeur de $\hat{\hat{x}}$ est mise à jour dans les deux cas avec la valeur de T. Comme $T.\sigma_{n+1}$ est égal à $T.\sigma'_{n+1}$ modulo L_S, on a bien $\hat{\hat{x}}.\sigma_{n+1}$ égal à $\hat{\hat{x}}.\sigma'_{n+1}$ modulo L_S. Si jamais on ne met pas à jour $\hat{\hat{x}}$, alors on conserve entre σ_{n+1} et σ'_{n+1} les valeurs qui étaient égal modulo L_S dans les états antérieurs.

On en conclut que le même comportement est possible dans les deux cas. S'il y a une mise à jour, on donne a x la valeur de T. On obtient donc bien deux états avec la valeur de $\hat{\hat{x}}$ égales modulo L_S.

1.B.2. On s'intéresse désormais au variables périodiques et donc aux prédicats $PeriodicUpdate$ et $PeriodicIdle$. On prouve tout d'abord que :

$$T.\sigma_n + 1 - \lfloor \frac{T.\sigma_n + 1 - \phi + J}{P} \rfloor * P - \phi = T.\sigma'_n + 1 - \lfloor \frac{T.\sigma'_n + 1 - \phi + J}{P} \rfloor * P - \phi$$

D'après les propriétés de la division euclidienne, il existe $k \in \mathbb{N}$ tel que :

$$T.\sigma_n + 1 - \phi + J = k * L_S + (T.\sigma_n + 1 - \phi + J) \pmod{L_S}$$

On a alors comme L_S est divisible par P :

$$\lfloor \frac{T.\sigma_n + 1 - \phi + J}{P} \rfloor = k * \frac{L_S}{P} + \lfloor \frac{(T.\sigma_n + 1 - \phi + J) \pmod{L_S}}{P} \rfloor$$

et donc :

$$\begin{array}{l} T.\sigma_n + 1 - \lfloor \frac{T.\sigma_n + 1 - \phi + J}{P} \rfloor * P - \phi = \\ (T.\sigma_n + 1 - \phi + J) \pmod{L_S} - \lfloor \frac{(T.\sigma_n + 1 - \phi + J) \pmod{L_S}}{P} \rfloor * P - J \end{array}$$

De même, on prouve que :

$$\begin{array}{l} T.\sigma'_n + 1 - \lfloor \frac{T.\sigma'_n + 1 - \phi + J}{P} \rfloor * P - \phi = \\ (T.\sigma'_n + 1 - \phi + J) \pmod{L_S} - \lfloor \frac{(T.\sigma'_n + 1 - \phi + J) \pmod{L_S}}{P} \rfloor * P - J \end{array}$$

Or comme $T.\sigma_n$ et $T.\sigma'_n$ sont égaux modulo L_S et donc on a :

$$(T.\sigma_n + 1 - \phi + J) \pmod{L_S} = (T.\sigma'_n + 1 - \phi + J) \pmod{L_S}$$

Donc on a bien :

$$T.\sigma_n + 1 - \lfloor \frac{T.\sigma_n + 1 - \phi + J}{P} \rfloor * P - \phi = T.\sigma'_n + 1 - \lfloor \frac{T.\sigma'_n + 1 - \phi + J}{P} \rfloor * P - \phi$$

De même, comme $\hat{\hat{x}}.\sigma_n$ et $ValMod(\hat{\hat{x}}.\sigma'_n, T.\sigma'_n, L_S)$ sont égaux modulo L_S, on peut prouver de la même manière que l'on a :

$$\lfloor \frac{T.\sigma_n+1-\phi+J}{P} \rfloor - \lfloor \frac{(\hat{\hat{x}}.\sigma_n-\phi+J) \pmod{L_S}}{P} \rfloor =$$
$$\lfloor \frac{T.\sigma'_n+1-\phi+J}{P} \rfloor - \lfloor \frac{(ValMod(\hat{\hat{x}}.\sigma'_n,T.\sigma'_n,L_S)-\phi+J) \pmod{L_S}}{P} \rfloor$$

$$T.\sigma_n - \lfloor \frac{T.\sigma_n+1-\phi+J}{P} \rfloor * P + \phi - J = T.\sigma'_n - \lfloor \frac{T.\sigma'_n+1-\phi+J}{P} \rfloor * P + \phi - J$$

$$\hat{\hat{x}}.\sigma_n - \lfloor \frac{T.\sigma_n+1-\phi+J}{P} \rfloor * P - \phi + J = ValMod(\hat{\hat{x}}.\sigma'_n, T.\sigma'_n, L_S) - \lfloor \frac{T.\sigma'_n+1-\phi+J}{P} \rfloor * P - \phi + J$$

De plus, étant donné que $T.\sigma_n$ et $T.\sigma'_n$ sont égaux modulo L_S et L_S est divisible par P, on a :

$$(\exists k \in \mathbb{N}, \exists \delta_k \in [-J..J] : T.\sigma_n + 1 = k * P + \phi + \delta_k) \Leftrightarrow$$
$$(\exists k \in \mathbb{N}, \exists \delta_k \in [-J..J] : T.\sigma'_n + 1 = k * P + \phi + \delta_k)$$

Ce sont ces différentes valeurs et l'existence de ces entiers k et δ_k qui définissent si les prédicats $PeriodicUpdate$ et $PeriodicIdle$ sont justes dans entre les états σ_n et σ_{n+1} et entre les états σ'_n et σ'_{n+1}. On peut donc avoir le même comportement dans les deux cas. Comme dans le cas d'une variable sporadique, entre σ_n et σ_{n+1} et entre σ'_n et σ'_{n+1} on met ou pas à jour la valeur de $\hat{\hat{x}}$ avec les valeurs de T mais dans les deux cas on a bien deux valeurs égales modulo L_S.

1.B.3. On s'intéresse désormais aux variables images d'une observations et aux prédicats des actions de ces variables définies par les propriétés des chemins.

Pour une observation $y \lesssim f(X)$, on construit dans l'état σ_{n+1} la valeur de la variable $H_{y \lesssim f(X)}$ à partir de la valeur de cette variable dans l'état σ_n. On sait que la variable $H_{y \lesssim f(X)}$ est équivalente dans les deux états σ'_n et σ_n, on montre que l'on peut construire une valeur de $H_{y \lesssim f(X)}$ en σ'_{n+1} équivalente à celle en σ_{n+1}

On construit la variable historique de l'état σ'_{n+1} en utilisant le même indice minimum que dans l'état σ_{n+1}. On prouve par la suite que cet indice minimum vérifie bien les propriétés du chemin. La construction de la variable historique est faite dans les deux systèmes par des actions similaires. Il s'agit premièrement de mettre à jour certaines caractéristiques contenues dans la variable historique dans l'état précédent pour les indices inférieurs à l'indice maximum.

Dans le système de transitions non réduit, entre l'état σ_n et l'état σ_{n+1} un ensemble de caractéristiques $\langle upd, dur, [z] : P \rangle$ de $H_{y \lesssim f(X)}.\sigma_n$ sont mises à jour, si on a :

$$\hat{\hat{z}}.\sigma_{n+1} = upd$$

De même, entre l'état σ'_n et l'état σ'_{n+1} l'ensemble de caractéristiques $\langle upd', dur, [z] : P \rangle$, tel que upd et upd' sont égaux modulo L_S, existe dans $H_{y \lesssim f(X)}.\sigma'_n$ et est mis à jour si on a :

$$\hat{\hat{z}}.\sigma'_{n+1} = upd'$$

Or comme $\hat{\hat{z}}.\sigma'_{n+1}$ est égal à $\hat{\hat{z}}.\sigma_{n+1}$ modulo L_S, qui est égal à upd qui est égal à upd' modulo L_S ces deux valeurs sont égales modulo L_S. D'après la proposition 52, elles

sont donc égales. Donc si une caractéristique est mise à jour entre σ_n et l'état σ_{n+1}, la caractéristique équivalente est aussi mise à jour entre σ'_n et l'état σ'_{n+1}.

La valeur dur est alors remplacé entre σ_n et l'état σ_{n+1} par $T.\sigma_{n+1} + 1 - upd$ et par $T.\sigma'_{n+1} + 1 - ValMod(upd, T.\sigma'_{n+1}, L_S)$ entre σ'_n et l'état σ'_{n+1}. D'après la proposition 51, les différences sont bien égales. On a bien deux ensembles de caractéristiques qui restent équivalents.

Les ensembles de caractéristiques qui ne sont pas mis à jour étaient déjà équivalents dans l'état précédent et le restent dans les états σ_n et σ'_n.

Les éléments placés à l'indice maximum de l'historique sont eux ajoutés et non mis à jour. Ils sont définis en fonction du temps et de la valeur du profil temporel des variables dans l'état courant. Dans le système de transitions non réduit, on a :

$$\begin{array}{l} H_{y \precsim f(X)}[IndexMax(H_{y \precsim f(X)}) + 1].\sigma_{n+1} = \\ \{\langle \hat{\hat{x}}.\sigma_{n+1}, T.\sigma_{n+1} + 1 - \hat{\hat{x}}.\sigma_{n+1}, [x, y]\rangle | x \in X\} \cup \\ \left\{ o \;\middle|\; \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \precsim g(Z)}.\sigma_{n+1}) \\ \wedge\, o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \wedge \nexists P' : P = [y] : P' \end{array} \right\} \end{array}$$

Dans le système de transitions réduit, on a :

$$\begin{array}{l} H_{y \precsim f(X)}[IndexMax(H_{y \precsim f(X)}) + 1].\sigma'_{n+1} = \\ \{\langle \hat{\hat{x}}.\sigma'_{n+1}, T.\sigma'_{n+1} + 1 - ValMod(\hat{\hat{x}}.\sigma'_{n+1}, T.\sigma'_{n+1}, L_S), [x, y]\rangle | x \in X\} \cup \\ \left\{ o \;\middle|\; \begin{array}{l} x \in X \wedge \exists Z \in \mathcal{V}ar(\mathcal{A}rchi), \exists g : \exists o' \in min(H_{x \precsim g(Z)}.\sigma'_{n+1}) \\ \wedge\, o' = \langle upd, dur, P\rangle \wedge o = \langle upd, dur, P : [y]\rangle \wedge \nexists P' : P = [y] : P' \end{array} \right\} \end{array}$$

Étant donné que l'on a d'après la proposition 51, comme le couple $(T.\sigma'_{n+1}, \hat{\hat{x}}.\sigma_{n+1})$ est égal au couple $(T.\sigma'_{n+1}, \hat{\hat{x}}.\sigma'_{n+1})$ modulo L_S et comme $T.\sigma_{n+1} + 1 - \hat{\hat{x}}.\sigma_{n+1}$ est borné par L_S, on a :

$$T.\sigma'_{n+1} + 1 - ValMod(\hat{\hat{x}}.\sigma'_{n+1}, T.\sigma'_{n+1}, L_S) = T.\sigma_{n+1} + 1 - \hat{\hat{x}}.\sigma_{n+1}$$

Comme les historiques des autres observations sont équivalents, on a bien les éléments au rang maximum qui sont équivalents dans les deux états σ_n et σ'_n.

1.B.4. On vérifie que si les éléments du minimum de cet historique vérifie les propriétés des chemins dans l'état σ_{n+1}, alors ces propriétés sont bien vérifiées dans l'état σ'_{n+1}. On peut donc bien avoir une variable historique dans l'état σ'_{n+1} qui a la même taille que dans l'état σ_n. Les propriétés d'un chemin sont définies d'après le domaine d'appartenance de dur ($DurationInterval$)et au domaine d'appartenance de T à un intervalle définit en fonction de dur. La vérification des prédicats dépend alors des valeurs de $T.\sigma_{n+1} - upd$ et de dur.

Dans l'état σ'_{n+1}, ces mêmes prédicats dépendent des valeurs de la différence $T.\sigma'_{n+1} - ValMod(upd', T.\sigma'_{n+1}, L_S)$ et de dur'. D'après l'équivalence des historiques et l'égalité des différences, si ces prédicats sont vérifiées en un σ_{n+1} alors ils sont donc aussi vérifiées en σ'_{n+1}.

Les conditions sur la durée qu'elles soient d'anticipation ou qu'elles soient vérifiées directement dans le cas d'une occurrence finie, sont définies par l'intervalle $DurationInterval$. Dans les deux cas, la définition de cette intervalle dépend des propriétés de la spécification et des différences entre T et upd où entre T et $ValMod(upd, T, L_S)$. Ces différences sont égales et donc les conditions sur la durée dur sont équivalentes.

On en déduit donc que l'on peut construire un état σ'_{n+1} équivalent à σ_{n+1}. Par récurrence, on construit donc une exécution σ' équivalente à l'exécution σ.

2. Il s'agit maintenant de prouver qu'à partir d'une exécution du système de transitions réduit, on peut construire une exécution équivalente du système de transitions non réduit. La preuve est symétrique à la preuve dans l'autre sens. On s'appuie ici aussi sur le fait que L_S borne dans les deux exécutions les différences entre T et les valeurs représentant des dates de mise à jour ou la valeur ValMod. Les différences sont alors égales au cours des deux exécutions. On peut donc construire une exécution équivalente.

□

7.2.5 *Complexité du système de transitions réduit*

On donne une borne supérieure sur la complexité du système de transitions réduit et donc la la taille du système et la taille du domaine de définition de chaque variables du système. Les différentes variables du système réduit sont le temps T, les profils temporels de chaque variable de l'architecture et les variables historiques de chaque relation d'observation.

Proposition 55. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification. Le domaine de* T *dans le système de transitions réduit est de taille* $2 * L_S$.

Proposition 56. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification. Pour toute variable* x *de l'architecture, le domaine du profil temporel* $\hat{\hat{x}}$ *de* x *dans le système de transitions réduit est de taille* $2 * L_S$ *et le domaine de la variable stabilité* $d_{\bar{x}}$ *est de taille* L_S.

Proposition 57. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et* L_S *la longueur de l'intervalle d'analyse de cette spécification. Pour toute observation* $y \lessdot f(X)$ *de l'architecture, le domaine de l'historique* $H_{y \lessdot f(X)}$ *dans le système de transitions réduit est de taille bornée par :*

$$2 * L_S * L_S^{2 * L_S * P}$$

où P *est le nombre de chemins de l'architecture (*$P = \mathtt{card}(\mathcal{P}aths(\mathcal{A}rchi))$*).*

Démonstration. Dans l'état initial, la variable T est initialisée à 0. Elle croît par la suite jusque la valeur $2 * L_S - 1$. Lorsque cette valeur est atteinte, la valeur L_S est assignée à T et elle croît ensuite de nouveau jusque $2 * L_S - 1$. Elle ne dépasse donc jamais cette valeur et son domaine de définition est donc $[0..2 * L_S - 1[$ de taille $2 * L_S$.

Pour une variable x, les valeurs de son profil temporel $\hat{\hat{x}}$ sont définies par les valeurs de T. Le domaine de définition de $\hat{\hat{x}}$ est donc le même que celui de T soit $[0..2 * L_S - 1[$ de taille $2 * L_S$. Concernant la stabilité $d_{\bar{x}}$ de x, la proposition 45 page 142 nous indique que toutes les variables de l'architecture sont sporadiques avec comme borne supérieure leur temps de sortie du régime initial strictement inférieur à la longueur de l'intervalle d'analyse L_S. On en déduit donc que la valeur de la stabilité est dans l'intervalle $[0..L_S[$ de taille L_S.

Concernant l'historique d'une observation $y \lessdot f(X)$, on considère un état σ_i d'une exécution définie par le système de transitions réduit.

On borne tout d'abord la taille du tableau c'est à dire le nombre d'indice. Pour chaque état d'une exécution σ' de la spécification, l'indice minimum de l'historique est défini comme le maximum des indices des états pointés par l'horloge de l'observation $y \lessdot f(X)$. L'indice courant définissant l'indice maximum du tableau, pour un état σ'_i il existe donc

une horloge c de l'observation tel que l'historique a pour taille $i - c(i) + 1$. Il y a autant d'états entre cet état et l'état actuel que de temps écoulé et donc cette taille est égale à $T.\sigma'_i - T.\sigma'_{c(i)} + 1$. La différence entre $T.\sigma'_i - T.\sigma'_{c(i)}$ est bornée tout au long de l'exécution par $L_S - 1$, la taille de l'historique est donc bornée par L_S dans les exécutions définies par la spécification. Par équivalence des exécutions, la taille de l'historique dans les exécutions du système de spécification réduit est aussi bornée par L_S dans chaque état.

On considère que la valeur de T dans l'état σ_i est connue. L'historique peut être alors défini comme une fonction qui à chaque indice associe un ensemble de triplets. Cet ensemble de triplets peut lui même être défini comme une fonction qui à chaque chemin de l'architecture associe un couple de valeurs représentant une date de mise à jour et la durée d'une occurrence. Les valeurs des dates de mise à jour sont dans l'intervalle $[0..2 * L_S - 1[$ mais il n'y a en réalité que L_S valeurs différentes possibles, la valeur de T étant fixée. En effet, d'après la proposition 52, deux valeurs égales modulo L_S dans cet état σ_i sont égales. Il existe donc un intervalle I de taille L_S et comprenant $T.\sigma_i$ dans lequel les dates de mise à jour des triplets prennent valeur. La durée d'une occurrence peut prendre L_S valeurs possibles, sa valeur étant déduite des variables stabilité des variables de l'architecture. Le couple "date de mise à jour, durée d'une occurrence" peut donc prendre $L_S * L_S$ valeurs possibles. Il existe donc une horloge c de l'observation $y \preccurlyeq f(X)$ tel que l'historique est donc pour chaque valeur de T une fonction de type :

$$[c(i)..i] \to P \to I \times [0..L_S - 1]$$

où P est le nombre de chemins de l'architecture($P = card(\mathcal{P}aths(\mathcal{A}rchi))$ et où les tailles des intervalles $[c(i)..i]$ et I sont inférieures à L_S. On en déduit que pour une valeur de T, le nombre de valeurs possibles de l'historique d'une observation est borné par $(L_S^2)^{L_S * P}$. Pour un état, T peut prendre $2 * L_S$ valeurs différentes, le nombre de valeurs possibles pour l'historique des valeurs est borné par $2 * L_S * (L_S^2)^{L_S * P}$. □

8

EXEMPLE DE SPÉCIFICATION

8.1 PRÉSENTATION

On illustre les travaux effectués dans cette thèse dans le cadre d'un exemple simple. Il s'agit de spécifier l'architecture et les propriétés d'un système régulateur de vitesse. Ce système contrôle la vitesse d'un véhicule afin de la stabiliser à une valeur choisie par le conducteur.

On donne ici une version simplifiée d'un tel régulateur. On considère que le système est défini par quatre composants :

- un capteur de la vitesse courante du véhicule ;
- le micromoteur de commande des gaz ;
- un calculateur ;
- la commande du conducteur.

La commande du conducteur est utilisée par celui ci pour donner la vitesse de croisière du véhicule. Le calculateur compare cette vitesse avec la vitesse courante mesurée. Il déduit de cette différence la valeur à appliquer à la commande des gaz qui contrôlent l'accélération du véhicule et donc agit sur sa vitesse. Finalement, un bus de communication permet la communication entre ces différents composants, le capteur de vitesse se trouvant par exemple sur une des roues du véhicule et la commande des gaz au sein du moteur.

L'architecture du système est illustrée figure 19. Les boites représentent les composants du système et les flèches la propagation des valeurs des variables le long du bus de communication. On introduit trois variables :

- `current` : la vitesse courante du véhicule
- `cruise` : la vitesse de croisière du véhicule
- `throttle` : la valeur à appliquer à la commande des gaz

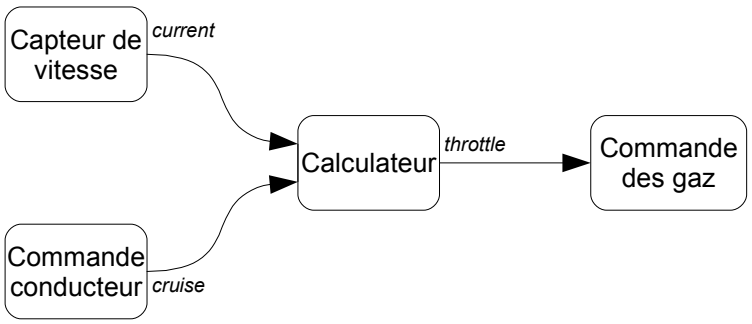

FIG. 19: Architecture de l'exemple

- communications :

$$\begin{array}{c} \text{`current} \lessdot \text{current} \\ \text{`cruise} \lessdot \text{cruise} \\ \text{`throttle} \lessdot \text{throttle} \end{array}$$

- calcul :

$$\text{throttle} \lessdot \text{calc}(\text{`current}, \text{`cruise})$$

FIG. 20: Spécification de l'architecture de l'exemple

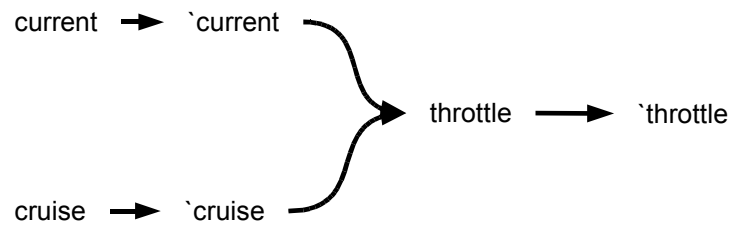

FIG. 21: Graphe défini par l'architecture de l'exemple

8.2 SPÉCIFICATION

On commence par spécifier l'architecture du système. On a une fonction calc calculant la valeur de la variable throttle à partir de celle de cruise et current ou plutôt à partir des copies de leur valeurs qui sont communiquées. On introduit trois nouvelles variables 'current, 'cruise, 'throttle représentant ces copies respectives des variables current, cruise, throttle à travers le bus de communication.

On considère ici que la lecture des entrées de la fonction calc est synchrone. L'architecture du système est alors spécifiée par quatre relations d'observations données figure 20. Le graphe associé à cette architecture est donné figure 21.

Afin de compléter la spécification, on donne les propriétés temps réel du système. On donne tout d'abord chacune de ces propriétés informellement avant de donner la spécification correspondante.

On commence par les propriétés sur le comportement de chacun des variables de l'architecture.

- la variable current est soumise à un prédicat de sporadicité avec deux bornes :
  - une borne inférieure donnant le temps minimum pour pouvoir calculer une valeur correcte de cette valeur ;
  - une borne supérieure pour que la valeur de cette vitesse soit mise à jour suffisamment souvent pour rester cohérente avec l'évolution de la vitesse réelle du véhicule ;
- la variable cruise est soumise à un prédicat de sporadicité avec une borne inférieure, il s'agit du temps minimum entre deux modifications de la vitesse par le conducteur ;
- pour les autres variables, on ne donne pas de propriété sur leur comportement, les propriétés sur ces variables sont exprimées par rapport aux chemins dont elles sont images.

Il existe différents chemins dans ce système, on donne des propriétés sur certains de ces chemins :

- comportement des variables :

$$\text{current}\ \{\text{Sporadic}(\delta_1, \Delta_1)\}$$
$$\text{cruise}\ \{\text{Sporadic}(\delta_2, +\infty)\}$$

- propriétés des communications :

$$[\text{current}, \text{'current}]\ \{\text{Lag}(\delta_3, +\infty)\}$$
$$[\text{cruise}, \text{'cruise}]\ \{\text{Lag}(\delta_3, +\infty)\}$$
$$[\text{throttle}, \text{'throttle}]\ \{\text{Lag}(\delta_3, +\infty)\}$$

- propriétés des calculs :

$$[\text{'current}, \text{throttle}]\ \{\text{Lag}(\delta_4, +\infty)\}$$
$$[\text{'cruise}, \text{throttle}]\ \{\text{Lag}(\delta_4, +\infty)\}$$

- propriétés des chemins de bout en bout :

$$[\text{current}, \text{'current}, \text{throttle}, \text{'throttle}]\ \{\text{Latency}(0, \Delta_5)\}$$
$$[\text{cruise}, \text{'cruise}, \text{throttle}, \text{'throttle}]\ \{\text{Shift}(0, \Delta_6)\}$$

FIG. 22: Spécification des propriétés temporelles de l'exemple

– pour chacun des chemins représentant une communication, c'est-à-dire les chemins $[\text{current}, \text{'current}]$, $[\text{cruise}, \text{'cruise}]$ et $[\text{throttle}, \text{'throttle}]$, on donne une propriété de lag avec une borne inférieure. Cette propriété spécifie donc qu'il n'est pas possible lorsque la source du chemin est mise à jour que la nouvelle occurrence de cette source soit disponible pour une mise à jour de l'image avant qu'une durée égale à cette borne inférieure se soit écoulée ;
– pour les chemins représentant un calcul, c'est-à-dire les chemins $[\text{'current}, \text{throttle}]$ et $[\text{'cruise}, \text{throttle}]$, on donne une propriété de lag avec une borne inférieure. Cette propriété donne le temps de calcul minimum et donc le temps minimum pour qu'une nouvelle occurrence des sources soit utilisée pour calculer une nouvelle occurrence de l'image ;
– pour le chemin $[\text{current}, \text{'current}, \text{throttle}, \text{'throttle}]$, on donne une propriété de latence avec une borne supérieure. Une occurrence de current est pertinente au moment de son apparition, moment où elle reflète la vitesse courante du véhicule. On limite donc le temps écoulé entre l'apparition de cette occurrence et les instants où elle est utilisée pour définir la valeur de 'throttle et donc définir l'accélération du véhicule ;
– pour le chemin $[\text{cruise}, \text{'cruise}, \text{throttle}, \text{'throttle}]$, on donne une propriété de décalage avec une borne supérieure. Ici une occurrence de cruise est pertinente tant que le conducteur ne l'a pas modifiée. La propriété de décalage permet de borner le temps entre les instants où cette occurrence n'a pas été remplacée et les instants où elle définit l'accélération du véhicule.

L'ensemble des propriétés temporelles est donné figure 22.

8.3 TEMPS DE SORTIE DU RÉGIME INITIAL

On donne maintenant le temps de sortie du régime initial de cette spécification. On vérifie pour cela que la spécification vérifie les conditions définies par la proposition 34 page 34 assurant d'avoir et de pouvoir calculer un temps de sortie du régime initial borné. Le graphe défini par l'architecture est bien à graphe à entrées élémentaires, cependant les deux autres propriétés ne sont pas satisfaites. En effet, pour le chemin $[\text{cruise}, \text{'cruise}, \text{throttle}]$, il n'y a ni prédicat de décalage, ni prédicat de latence caractérisant ce chemin et il n'y a pas non plus de sur-chemin soumis à un prédicat de latence.

- comportement des variables :

$$\text{current}\ \{\text{Sporadic}(\delta_1, \Delta_1)\}$$
$$\text{cruise}\ \{\text{Sporadic}(\delta_2, \Delta_2)\}$$

- propriétés des communications :

$$[\text{current}, \text{`current}]\ \{\text{Lag}(\delta_3, +\infty)\}$$
$$[\text{cruise}, \text{`cruise}]\ \{\text{Lag}(\delta_3, +\infty)\}$$
$$[\text{throttle}, \text{`throttle}]\ \{\text{Lag}(\delta_3, +\infty)\}$$

- propriétés des calculs :

$$[\text{`current}, \text{throttle}]\ \{\text{Lag}(\delta_4, +\infty)\}$$
$$[\text{`cruise}, \text{throttle}]\ \{\text{Lag}(\delta_4, +\infty)\}$$

- propriétés des chemins de bout en bout :

$$[\text{current}, \text{`current}, \text{throttle}, \text{`throttle}]\ \{\text{Latency}(0, \Delta_5)\}$$
$$[\text{cruise}, \text{`cruise}, \text{throttle}, \text{`throttle}]\ \{\text{Shift}(0, \Delta_6)\}$$

FIG. 23: Spécification ajustée des propriétés temporelles de l'exemple

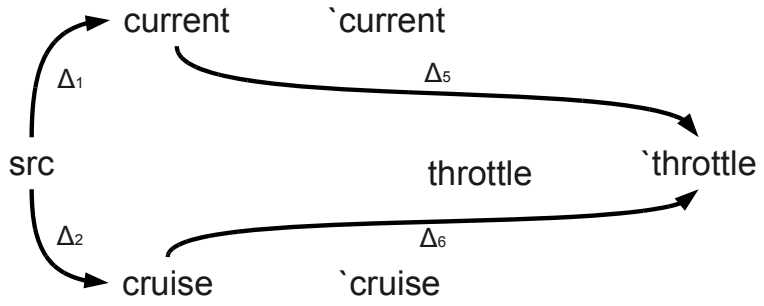

FIG. 24: Graphe de calcul du temps de sortie du régime initial

De plus, la variable cruise n'est pas image d'une observation et n'est pas soumise à un prédicat de sporadicité ayant une borne supérieure et n'est pas périodique.

On ajoute alors une propriété à la spécification permettant de remplir les conditions définies par la proposition 34. On spécifie une borne supérieure sur la sporadicité de cette variable. Il s'agit de donner une borne largement supérieure aux autre temps du système et n'interférant donc pas avec les autres propriétés.

Les conditions sur les chemins ne sont toujours pas satisfaites. Cependant, en utilisant la proposition 46 page 144, on a :

$$\left(\begin{array}{c}\text{cruise}\ \{\text{Sporadic}(\delta_2, \Delta_2)\} \wedge \\ [\text{cruise}, \text{`cruise}, \text{throttle}, \text{`throttle}]\ \{\text{Shift}(0, \Delta_6)\}\end{array}\right) \Rightarrow$$
$$[\text{cruise}, \text{`cruise}, \text{throttle}, \text{`throttle}]\ \{\text{Latency}(0, \Delta_2 + \Delta_6)\}$$

Cette propriété de latence permet finalement de satisfaire les conditions de calcul d'un temps de sortie du régime initial borné. On définit donc un premier graphe de calcul qui est donné figure 24 et construit à partir de la propriété déduite précédemment et des propriétés de la spécification.

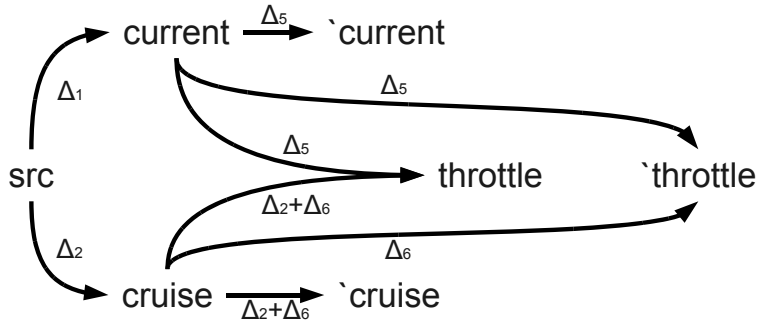

FIG. 25: Graphe de calcul du temps de sortie du régime initial

Dans ce graphe, il existe trois variables pour lesquelles il n'y a pas de chemin depuis les sources. La proposition 30 nous permet de déduire de la spécification des propriétés les propriétés de latence suivante :

$[\mathtt{current}, \mathtt{`current}]\ \{\mathtt{Latency}(0, \Delta_5)\}$
$[\mathtt{cruise}, \mathtt{`cruise}]\ \{\mathtt{Latency}(0, \Delta_2 + \Delta_6)\}$
$[\mathtt{current}, \mathtt{`current}, \mathtt{throttle}]\ \{\mathtt{Latency}(0, \Delta_5)\}$
$[\mathtt{cruise}, \mathtt{`cruise}, \mathtt{throttle}]\ \{\mathtt{Latency}(0, \Delta_2 + \Delta_6)\}$

On utilise ces propriétés pour compléter le graphe de calcul. Le graphe obtenu est donné figure 25.

De ce graphe, on déduit tout d'abord le temps de sortie du régime initial des variables.

$\Delta_{\mathtt{current}} = \Delta_1$
$\Delta_{\mathtt{cruise}} = \Delta_2$
$\Delta_{\mathtt{`current}} = \Delta_1 + \Delta_5$
$\Delta_{\mathtt{`cruise}} = 2 * \Delta_2 + \Delta_6$
$\Delta_{\mathtt{throttle}} = \min(\Delta_1 + \Delta_5, 2 * \Delta_2 + \Delta_6)$
$\Delta_{\mathtt{`throttle}} = \min(\Delta_1 + \Delta_5, \Delta_2 + \Delta_6)$

On donne ensuite le temps de sortie du régime initial des chemins correspondant à un arc du graphe. Pour ces chemins, le temps de sortie est défini par le choix 4 page 89.

$\Delta_{[\mathtt{current},\mathtt{`current}]} = \Delta_1 + \Delta_5$
$\Delta_{[\mathtt{cruise},\mathtt{`cruise}]} = 2 * \Delta_2 + \Delta_6$
$\Delta_{[\mathtt{current},\mathtt{`current},\mathtt{throttle}]} = \Delta_1 + \Delta_5$
$\Delta_{[\mathtt{cruise},\mathtt{`cruise},\mathtt{throttle}]} = 2 * \Delta_2 + \Delta_6$
$\Delta_{[\mathtt{current},\mathtt{`current},\mathtt{throttle},\mathtt{`throttle}]} = \Delta_1 + \Delta_5$
$\Delta_{[\mathtt{cruise},\mathtt{`cruise},\mathtt{throttle},\mathtt{`throttle}]} = \Delta_2 + \Delta_6$

Finalement, on déduit le temps de sortie du régime initial des chemins restant d'après le temps de sortie du régime initial de leurs sur-chemins, conformément au choix 5 page 90.

$\Delta_{[\mathtt{throttle},\mathtt{`throttle}]} = \min(\Delta_1 + \Delta_5, \Delta_2 + \Delta_6)$
$\Delta_{[\mathtt{`current},\mathtt{throttle}]} = \Delta_1 + \Delta_5$
$\Delta_{[\mathtt{`cruise},\mathtt{throttle}]} = \Delta_2 + \Delta_6$
$\Delta_{[\mathtt{`current},\mathtt{throttle},\mathtt{`throttle}]} = \Delta_1 + \Delta_5$
$\Delta_{[\mathtt{`cruise},\mathtt{throttle},\mathtt{`throttle}]} = \Delta_2 + \Delta_6$

Le temps de sortie du régime initial de la spécification est le maximum des temps de sortie du régime initial des chemins et des variables du système. Ce temps Δ_S est donc défini par :

$$\Delta_S = max\{\Delta_1 + \Delta_5, 2 * \Delta_2 + \Delta_6\}$$

8.4 SYSTÈME DE TRANSITIONS FINI

On choisit tout d'abord une longueur de l'intervalle d'analyse. Étant donné qu'il n'y a pas de variables périodiques, ce choix est déduit du temps de sortie du régime initial de la spécification. On choisit donc L_S la longueur de l'intervalle d'analyse égal à $\Delta_S + 1$.

$$A_{current} \triangleq \left(\begin{array}{c} \hat{current}' = T' \wedge SporadicUpdate(T, ValMod(\hat{current}, T, L_S), \delta_1, \Delta_1) \\ \vee \\ \hat{current}' = \hat{current} \wedge SporadicIdle(T, ValMod(\hat{current}, T, L_S), \delta_1, \Delta_1) \end{array} \right)$$

$$A_{cruise} \triangleq \left(\begin{array}{c} \hat{cruise}' = T' \wedge SporadicUpdate(T, ValMod(\hat{cruise}, T, L_S), \delta_1, \Delta_1) \\ \vee \\ \hat{cruise}' = \hat{cruise} \wedge SporadicIdle(T, ValMod(\hat{cruise}, T, L_S), \delta_1, \Delta_1) \end{array} \right)$$

$$A_{`current} \triangleq \text{Let } \langle upd, dur, [current, `current]\rangle = min(H'_{current \preceq current}) \text{ in } \left(\begin{array}{c} upd = 0 \wedge T' < \Delta_{[z]:P} \\ \vee \\ upd \neq 0 \wedge \\ \left(\begin{array}{c} `\hat{current}' = T' \wedge T' \geq \delta_3 + ValMod(upd, T', L_S) \\ \vee `\hat{current}' = `\hat{current} \end{array} \right) \end{array} \right)$$

$$A_{`cruise} \triangleq \text{Let } \langle upd, dur, [cruise, `cruise]\rangle = min(H'_{cruise \preceq cruise}) \text{ in } \left(\begin{array}{c} upd = 0 \wedge T' < \Delta_{[cruise, `cruise]} \\ \vee \\ upd \neq 0 \wedge \\ \left( \begin{array}{c} `\hat{cruise}' = T' \wedge T' \geq \delta_3 + ValMod(upd, T', L_S) \\ \vee `\hat{cruise}' = `\hat{cruise} \end{array} \right) \end{array} \right)$$

$$\begin{array}{l} A_{throttle} \triangleq \\ \wedge \text{ Let } \langle upd, dur, [`current, throttle]\rangle \in min(H'_{throttle \preceq calc(`current, `cruise)}) \text{ in } \\ \left( \begin{array}{c} upd = 0 \wedge T' < \Delta_{[`current, throttle]} \\ \vee \\ upd \neq 0 \wedge \\ \left(\begin{array}{c} \hat{throttle}' = T' \wedge T' \geq \delta_4 + ValMod(upd, T', L_S) \\ \vee \hat{throttle}' = \hat{throttle} \end{array} \right) \end{array} \right) \\ \wedge \text{ Let } \langle upd, dur, [`cruise, throttle]\rangle \in min(H'_{throttle \preceq calc(`current, `cruise)}) \text{ in } \\ \left( \begin{array}{c} upd = 0 \wedge T' < \Delta_{[`cruise, throttle]} \\ \vee \\ upd \neq 0 \wedge \\ \left(\begin{array}{c} \hat{throttle}' = T' \wedge T' \geq \delta_4 + ValMod(upd, T', L_S) \\ \vee \hat{throttle}' = \hat{throttle} \end{array} \right) \end{array} \right) \end{array}$$

$$
\begin{array}{l}
A_{`throttle} \triangleq \\
\wedge \text{ Let } \langle upd, dur, [`throttle, throttle]\rangle \in min(H'_{throttle \preceq throttle}) \text{ in} \\
\left(\begin{array}{c}
upd = 0 \ \wedge \ T' < \Delta_{[`throttle,throttle]} \\
\vee \\
upd \neq 0 \ \wedge \\
\left(\begin{array}{c} \hat{`throttle}' = T' \ \wedge T' \geq \delta_3 + ValMod(upd, T', L_S) \\ \vee \ \hat{`throttle}' = \hat{`throttle} \end{array}\right)
\end{array}\right) \\
\wedge \text{ Let } \langle upd, dur, [current, `current, throttle, `throttle]\rangle \in min(H'_{throttle \preceq throttle}) \text{ in} \\
\left(\begin{array}{c}
upd = 0 \ \wedge \ T' < \Delta_{[current,`current,throttle,`throttle]} \\
\vee \\
upd \neq 0 \ \wedge \\
\left(\left(\begin{array}{c} \hat{`throttle}' = T' \ \wedge \\ T' < ValMod(upd, T', L_S) + \Delta_5 \end{array}\right)\right) \\
\vee \\
\left(\left(\begin{array}{c} \hat{`throttle}' = \hat{`throttle} \wedge \\ T' < ValMod(upd, T', L_S) + \Delta_5 \end{array}\right)\right)
\end{array}\right) \\
\wedge \text{ Let } \langle upd, dur, [cruise, `cruise, throttle, `throttle]\rangle \in min(H'_{throttle \preceq throttle}) \text{ in} \\
\left(\begin{array}{c}
upd = 0 \ \wedge \ T' < \Delta_{[cruise,`cruise,throttle,`throttle]} \\
\vee \\
upd \neq 0 \ \wedge \\
\left(\left(\begin{array}{c} \hat{`throttle}' = T' \ \wedge \\ dur \geq T' - ValMod(upd, T', L_S) - \Delta_6 + 2 \end{array}\right)\right) \\
\vee \\
\left(\left(\begin{array}{c} \hat{`throttle}' = \hat{`throttle} \wedge \\ dur \geq T' - ValMod(upd, T', L_S) - \Delta_6 + 2 \end{array}\right)\right)
\end{array}\right)
\end{array}
$$

On ne donne pas ici les actions décrivant les conditions d'anticipation. Comme il n'y a pas de propriétés de pérennité ou de propriété de stabilité, celles ci sont inutiles. On ne précise pas non plus les actions construisant l'historique des observations, cette action ne varie pas selon les propriétés du système.

Finalement, la relation de transition $\rightarrow$ du système est définie par :

$$
\begin{array}{l}
\sigma_i \rightarrow \sigma_{i+1} \vDash \\
\qquad \left(\begin{array}{l} T.\sigma_{i+1} = T.\sigma_i + 1 - L_S \text{ si } T.\sigma_i + 1 \geq 2 * L_S \\ T.\sigma_{i+1} = T.\sigma_i + 1 \text{ sinon} \end{array}\right) \wedge \\
\qquad A_{current}.\sigma_i.\sigma_{i+1} \ \wedge \ A_{cruise}.\sigma_i.\sigma_{i+1} \ \wedge \ A_{`current}.\sigma_i.\sigma_{i+1} \ \wedge \\
\qquad A_{`cruise}.\sigma_i.\sigma_{i+1} \ \wedge \ A_{throttle}.\sigma_i.\sigma_{i+1} \ \wedge \ A_{`throttle}.\sigma_i.\sigma_{i+1}
\end{array}
$$

L'état initial est donnée par la définition 78 page 78.

8.5 RÉSULTAT D'ANALYSE

L'analyse de ce système a été réalisée automatiquement par deux outils, Spin et TLA+. Une présentation et une explication sur l'utilisation de ces deux outils pour l'analyse d'une spécification est donnée dans l'annexe B, le problème étant de traduire le système de transition finie équivalent à une spécification dans la syntaxe utilisée par ces outils. Les deux outils permettent de déterminer la satisfiabilité de la spécification si

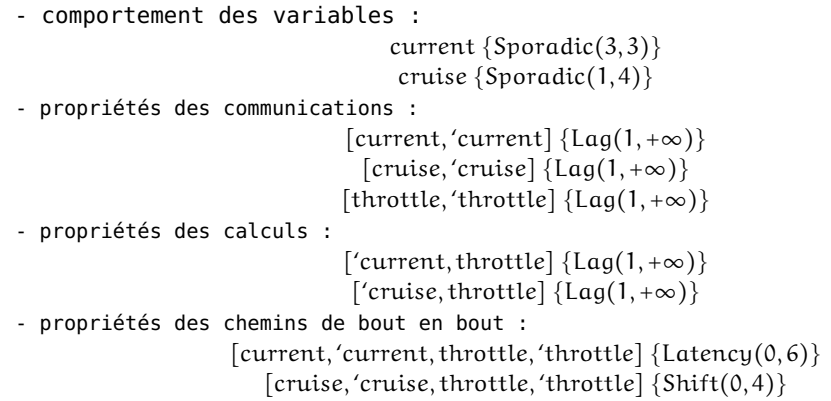

FIG. 26: Spécification satisfiable

les paramètres de cette spécification ne génèrent pas un nombre trop important d'états. Les résultats donnés ici sont obtenus en utilisant l'outil Spin.

Comme étudié dans le paragraphe 7.2.5 page 160, le nombre d'états dépend de la longueur de l'intervalle d'analyse et du nombre de chemins de propagation et donc d'observations. Comme on ne veut pas modifier l'architecture de la spécification, les paramètres de la spécification permettent uniquement de modifier la longueur de l'intervalle d'analyse. En pratique, on observe que l'on a des résultats pour des périodes d'analyse de longueur inférieure à 15, ce qui correspond à des périodes de l'ordre de 4 pour une latence de l'ordre de 10. Une spécification dont l'analyse a prouvé la satisfiabilité est par exemple donnée figure 26

Afin de réduire le nombre d'état à explorer, il est possible de compléter la spécification par des propriétés déduites des propriétés de la spécification données par l'utilisateur. Le but est d'éviter à l'outil d'explorer des séquences d'états qui ne vérifient pas la spécification et mènent systématiquement à une situation d'interblocage. Ainsi une possibilité est de limiter la sporadicité des autres variables que les entrées du système. Les paramètres de ces propriétés de sporadicité peuvent être déduits de propositions telle que la proposition 45 page 142. On a alors une spécification telle que celle donnée figure 27.

Rajouter de telles propriétés permet d'augmenter sensiblement la longueur de l'intervalle d'analyse en diminuant le nombre d'états effectivement construits par l'outil d'exploration. Néanmoins dans tous les cas, la longueur de cet intervalle est trop petite pour étudier des spécifications comparables à un système réel. En effet, il est possible de modifier l'échelle des temps, c'est-à-dire changer la valeur représentée par chaque unité de temps. Cependant, exprimée dans cette échelle, une période aussi petite que les périodes analysées ne permet pas d'avoir des écarts relatifs entre les différents paramètres suffisamment précis. Il n'est pas possible par exemple de spécifier que le minimum des propriétés de lag des différentes communications est égal au dixième du maximum de la propriété de sporadicité de la variable $current$.

Une remarque sur l'utilisation de l'outil Spin est sa convergence rapide dans le cas d'une spécification satisfiable. Dans le cas d'une spécification non satisfiable, par exemple en mettant une borne maximum sur la propriété de latence trop faible par rapport au décalage créé par les propriétés de lag le long des chemins de propagation, l'outil a besoin de temps pour affirmer la non satisfiabilité de la spécification. En effet,

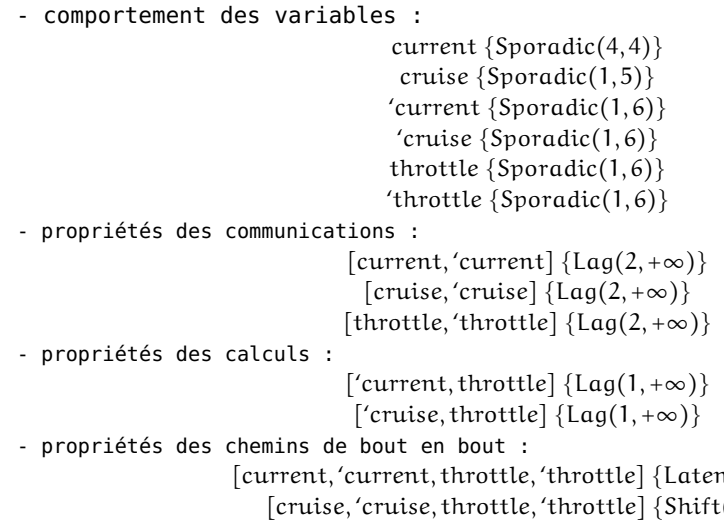

FIG. 27: Spécification satisfiable

dans le cas d'une spécification satisfiable, il s'agit de trouver une exécution infinie vérifiant cette satisfiabilité. L'outil s'arrête dès qu'une telle exécution est trouvée. Dans le cas d'une spécification non satisfiable, l'outil se doit d'étudier toutes les séquences d'états définies par la spécification, cette exhaustivité entraînant une réponse plus tardive de l'outil.

9 CONCLUSION

9.1 RÉSULTATS

Dans le contexte des systèmes temps réel, le but de ce travail a été de donner le moyen de spécifier les propriétés temps réel d'un système comme des propriétés des données du système et non des mécanismes. Les propriétés temps réel d'un système sont alors exprimées comme des conditions de validité des valeurs prises par les variables et propagées dans le système. Ces conditions de validité dépendent de la signification des valeurs prises par ces variables mais aussi des dépendances entre les valeurs d'une variable et les valeurs des autres variables du système.

Afin d'exprimer ces propriétés, on a premièrement introduit des variables permettant de caractériser le comportement d'une variable applicative du système au cours d'une exécution. Pour cela, on a défini la notion d'occurrence d'une variable et introduit une variable, le profil temporel, donnant les dates de mise à jour d'une variable applicative, c'est-à-dire les instants de changement d'occurrence.

Pour étudier les dépendances entre les valeurs ou plutôt les occurrences des variables du système, on décrit l'architecture d'un système par un ensemble de relations d'observation. La relation d'observation est restreinte ici à une relation entre une variable image, une fonction et un ensemble de variables sources. Les relations d'observation modélisent les calculs et communications du système et décrivent à la fois une relation entre les valeurs des variables sources et images et le décalage introduit par les calculs et les communications modélisés. A partir d'une architecture définie comme un ensemble de relations d'observation, on construit les chemins de propagation du système. Ces chemins définissent les dépendances qui existent entre les occurrences des différentes variables du système. Une occurrence de l'image d'un chemin dépend d'une occurrence de la source de ce chemin.

Deux types de propriétés temps réel du système sont définis. Premièrement, on restreint le comportement temporel d'une variable en caractérisant le temps écoulé entre deux mises à jour. Ensuite, pour chaque chemin de propagation de l'architecture, on donne des propriétés définissant si en un instant une occurrence de la source d'un chemin peut être utilisée pour définir une occurrence de l'image de ce chemin. La validité d'une occurrence d'une source du chemin est exprimée sous la forme de bornes que doivent vérifier des différences entre valeurs caractérisant le comportement temporel de la source, de l'image et le décalage introduit le long du chemin.

La spécification d'un système est donc définie par un couple $\langle \mathcal{A}rchi, \mathcal{P}aths \rangle$ décrivant l'architecture du système et un ensemble de propriétés temps réel sur les variables et les chemins de propagation de cette architecture. Pour étudier l'ensemble des exécutions définies par la sémantique opérationnelle de la spécification, on définit un modèle opérationnel, sous la forme d'un système de transitions, définissant un ensemble d'exécutions équivalent à la sémantique opérationnelle d'une spécification définie à base d'observations. Pour cela, on reformule tout d'abord les propriétés d'une spécification sous une forme équivalente. On définit ensuite, à partir de ces reformulations, un système de transitions équivalent à celui défini par une spécification mais étant un système à état fini. Ce système permet d'étudier la satisfiabilité d'une spécification.

La prochaine étape est de permettre de vérifier la validité de l'implantation d'une spécification. Il s'agit alors de vérifier que les mécanismes utilisés pour implanter les communications et les calculs modélisés par les relations d'observation de l'architecture vérifient la spécification. Une approche possible étudiée dans [LMPQ09] est de distinguer différentes étapes dans la propagation des occurrences des sources d'une observation vers l'image. Ces étapes doivent être définies selon le mécanisme d'implantation de l'observation et restreignent la disponibilité des occurrences présentes dans les variables historiques pour mettre à jour l'image de cette observation. Par exemple, pour une observation modélisant une communication, les occurrences de l'historique sont disponibles pour mettre à jour une image uniquement lorsque l'événement de réception d'un message contenant ces occurrences s'est produit. La réalisation de ces étapes dépend des propriétés des mécanismes implantés. Une possibilité est donc de restreindre le comportement du système de transitions réduit équivalent à la spécification selon les propriétés temporelles des mécanismes et définir ainsi un nouveau système de transitions. Une étude de satisfiabilité d'un tel système doit démontrer que malgré la disponibilité réduite des occurrences des sources, une occurrence des sources vérifiant les propriétés de la satisfaction est toujours disponible pour définir l'occurrence de l'image. La satisfiabilité de ce système démontre donc la satisfaction de la spécification par l'implantation.

On s'intéresse aussi à la génération d'une implantation. Il s'agit de définir dans quels états les étapes de la propagation des occurrences sources peuvent être franchies. Pour cela, on veut utiliser un système de transitions semblable au système utilisé pour la vérification d'une implantation. Le franchissement des étapes de propagation des occurrences est cependant indéterministe. La difficulté est alors de déterminiser le franchissement de ces étapes par un algorithme de génération de contrôleur. Les conditions de franchissement des étapes de propagation données par les contrôleurs générés doivent être utilisées pour définir une implantation vérifiant la spécification.

Il est aussi intéressant d'étudier les liens entre les propriétés temps réel d'une spécification à base d'observation et une expression classique de propriétés temps réel, par exemple sous la forme d'échéances sur les opérations d'un système. On cherche par exemple à prouver que le respect d'un ensemble d'échéances sur les opérations du système implique le respect d'une spécification à base d'observations. L'existence d'une telle implication permet alors de vérifier les propriétés de la spécification par une analyse d'ordonnancement. On a alors la possibilité d'utiliser les résultats et outils de ce domaine.

Finalement, dans le cadre de la conception de système dirigée par les modèles, il est sans doute possible d'étendre la sémantique du langage de modélisation UML et de ses profils adaptés au temps réel ou encore la sémantique du langage AADL avec les propriétés temporelles définies dans cette thèse.

[ABRW93] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Data consistency in hard real-time systems. Technical report, 1993. (Cité page 26.)

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :183–235, 1994. (Cité pages 33 et 35.)

[AF03] S. Anderson and J. K. Filipe. Guaranteeing temporal validity with a real-time logic of knowledge. In *ICDCSW '03 : Proceedings of the 23rd Int'l Conf. on Distributed Computing Systems*, pages 178–183. IEEE Computer Society, 2003. (Cité page 27.)

[AL94] M. Abadi and L. Lamport. An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems*, 16(5) :1543–1571, 1994. (Cité pages 37 et 39.)

[AMdS07] C. André, F. Mallet, and R. de Simone. Time Modeling in MARTE. In *Forum on specification & Design Languages (FDL)*, pages 268–273. ECSI, 2007. (Cité page 27.)

[AS85] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21(4) :181–185, 1985. (Cité page 30.)

[BCFS05] J. P. Bodeveix, D. Chemouil, M. Filali, and M. Strecker. Towards formalising aadl in proof assistants. *Electronic Notes in Theoretical Computer Science*, 141(3) :153–169, 2005. (Cité page 27.)

[BdS91] F. Boussinot and R. de Simone. The ESTEREL language. Research Report, INRIA, 1991. (Cité page 24.)

[Ben08] B. Ben Hedia. *Analyse temporelle des systèmes d'acquisition de données : une approche à base d'automates temporisés communicants et d'observateurs*. Thèse de doctorat, Insa Lyon, Lyon, 2008. (Cité pages 25 et 26.)

[BGL02] M. Bozga, S. Graf, and Mounier L. If-2.0 : A validation environment for component-based real-time systems. In *In Proceedings of Conference on Computer Aided Verification, CAV'02*, pages 343–348. Springer Verlag, 2002. (Cité page 25.)

[BJB05] B. Ben Hedia, F. Jumel, and J. P. Babau. Formal evaluation of quality of service for data acquisition systems. In *Forum on specification and Design Languages, FDL'05, Proceedings*, pages 579–589. ECSI, 2005. (Cité page 25.)

[BLRV07] B. Berthomieu, D. Lime, O. H. Roux, and F. Vernadat. Reachability problems and abstract state spaces for time petri nets with stopwatches. *Discrete Event Dynamic Systems*, 17(2) :133–158, 2007. (Cité page 36.)

[BML99] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee. Synthesis of embedded software from synchronous dataflow specifications. *Journal of VLSI Signal Processing Systems*, 21 :151–166, 1999. (Cité page 24.)

[BMN00] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1) :12–42, 2000. (Cité page 21.)

[BR08] M. Boyer and O. H. Roux. On the compared expressiveness of arc, place and transition time Petri nets. *Fundamenta Informaticae*, 88(3) :225–249, 2008. (Cité page 37.)

[BY04] J. Bengtsson and W. Yi. Timed automata : Semantics, algorithms and tools. pages 87–124. 2004. (Cité page 34.)

[CdA95] Jp. Courtiat, R.C. de Oliveira, and L. Andriantsiferana. Specification and validation of multimedia protocols using RT-Lotos. In *Proceedings 5th Workshop on Future Trends in Distributed Computing Systems*, pages 28–30. IEEE Press, 1995. (Cité page 32.)

[Cha97] M. Charpentier. *Assistance à la répartition des systèmes réactifs*. Thèse de doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, novembre 1997. (Cité pages 38, 42, 43, 47, et 60.)

[DB05] J. DeAntoni and J. P. Babau. A mda approach for systems dedicated to process control. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005)*, pages 567–570. IEEE Computer Society, 2005. (Cité page 25.)

[ED89] P. van Eijk and M. Diaz, editors. *Formal Description Technique Lotos : Results of the Esprit Sedos Project*. Elsevier Science Inc., 1989. (Cité page 32.)

[EMSS91] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *CAV '90 : Proceedings of the 2nd International Workshop on Computer Aided Verification*, pages 136–145. Springer-Verlag, 1991. (Cité page 22.)

[FGJ06] P. H. Feiler, D. P. Gluch, and Hudak J. J. The architecture analysis and design language (AADL) : An introduction. Technical report, CMU/SEI, 2006. (Cité page 27.)

[Fon01] C. Fong. Discrete-time dataflow models for visual simulation in Ptolemy II. Master's thesis, Electronics Research Laboratory, University of California, Berkeley, 2001. (Cité page 24.)

[FPY02] E. Fersman, P. Pettersson, and W. Yi. Timed automata with asynchronous processes : Schedulability and decidability. In *TACAS '02 : Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 67–82. Springer-Verlag, 2002. (Cité pages 34 et 35.)

[HCRP91] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language LUSTRE. In *Proceedings of the IEEE*, pages 1305–1320, 1991. (Cité page 24.)

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8) :666–677, 1978. (Cité page 32.)

[Hol03] G. Holzmann. *Spin model checker, the : primer and reference manual*. Addison-Wesley Professional, 2003. (Cité page 199.)

[Lam94] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3) :872–923, 1994. (Cité pages 37 et 103.)

[LL73] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment, 1973. (Cité page 23.)

[LMPQ09] T. Le Berre, P. Mauran, G. Padiou, and P. Quéinnec. A Data Oriented Approach for Real-Time Systems. In *International Conference on Real-Time and Network Systems (RTNS)*. IEEE Computer Society, 2009. (Cité page 174.)

[LPY97] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2) :134–152, October 1997. (Cité page 33.)

[LSV96] E.A. Lee and A. Sangiovanni-Vincentelli. Comparing models of computation. pages 234–241, 1996. (Cité page 39.)

[MBB09] L. Morel, J. P. Babau, and B. Ben Hedia. Formal modelling framework of data acquisition modules using a synchronous approach for timing analysis. In *Proceedings of the 30th IFAC Workshop on Real-Time Programming and 4th International Workshop on Real-Time Software*, pages 123–130, 2009. (Cité page 25.)

[Mer74] P. M. Merlin. *A study of the recoverability of computing systems.* PhD thesis, 1974. (Cité page 36.)

[Mil82] R. Milner. *A Calculus of Communicating Systems.* Springer-Verlag, 1982. (Cité page 32.)

[Ram74] C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. Technical report, Cambridge, MA, USA, 1974. (Cité page 36.)

[RR88] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58(1-3) :249–261, 1988. (Cité page 32.)

[SRL90] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols : An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9) :1175–1185, 1990. (Cité page 23.)

[Sta88] J.A. Stankovic. Misconceptions about real-time computing : a serious problem for next-generation systems. *Computer*, 21 :10–19, 1988. (Cité page 17.)

[TC94] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming - Euromicro Journal (Special Issue on Parallel Embedded Real-Time Systems)*, 40 :117–134, 1994. (Cité page 23.)

[The08] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V8.2*, 2008. `http://coq.inria.fr`. (Cité page 20.)

[Tho90] W. Thomas. Automata on infinite objects. pages 133–191. MIT Press, 1990. (Cité page 32.)

[XSS+96] M. Xiong, R. Sivasankaran, J. A. Stankovic, K. Ramamritham, and D. Towsley. Scheduling transactions with temporal constraints : exploiting data semantics. In *RTSS '96 : Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 240–253. IEEE Computer Society, 1996. (Cité page 26.)

[Yi91] W. Yi. CCS + time = an interleaving model for real time systems. In *Proceedings of the 18th international colloquium on Automata, languages and programming*, pages 217–228. Springer-Verlag, 1991. (Cité page 32.)

A

PROPRIÉTÉS DE COHÉRENCE

On présente dans cette annexe un ensemble de propriétés qui pourrait être ajouté à l'ensemble des propriétés temporelles d'une spécification.

A.1 COHÉRENCE D'UN COUPLE DE CHEMINS DE PROPAGATION

On définit une propriété sur les couples de chemins de propagation. On cherche à spécifier que la différence entre les décalages le long de deux chemins est bornée. Les occurrences des sources des chemins ont donc été prises dans des états temporellement proches.

Pour cela, de même que l'on caractérise les propriétés temporelles d'un chemin avec un ensemble de prédicats définis sur les caractéristiques temporelles du chemin, on utilise un ensemble de prédicats portant sur les caractéristiques temporelles de deux chemins.

Définition 92. *Cohérence d'un couple de chemins. Étant donné un ensemble d'observations d'occurrences* $\bar{\mathcal{O}}\mathrm{bs}$ *et deux chemins* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2 *bien définis par* $\bar{\mathcal{O}}\mathrm{bs}$*, la cohérence relative de ces chemins est paramétrée par un ensemble de prédicats* $\{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\}$ *si on a :*

$$\sigma \vDash P_1, P_2\ \{\mathtt{Predicate}_1(\delta_1, \Delta_1), \ldots, \mathtt{Predicate}_n(\delta_n, \Delta_n)\} \triangleq$$
$$\exists c_1 \in \mathcal{C}(P_1).\sigma, \exists c_2 \in \mathcal{C}(P_2).\sigma, \forall i \in \mathbb{N} : \begin{pmatrix} \mathtt{Predicate}_1(P_1, c_1, P_2, c_2, \delta_1, \Delta_1).\sigma_i \\ \wedge\ \ldots\ \wedge \\ \mathtt{Predicate}_n(P_1, c_1, P_2, c_2, \delta_n, \Delta_n).\sigma_i \end{pmatrix}$$

et où

$$\forall i \in [1..n] : \mathtt{Predicate}_i \in \{\mathtt{Lag}, \mathtt{Shift}\}$$

On donne la définition des deux prédicats utilisés.

Définition 93. *Décalage. Pour une exécution* σ*, le prédicat de décalage pour un état* σ_i *et pour deux chemins* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2 *est défini par :*

$$\mathtt{Shift}(P_1, c_1, P_2, c_2, \delta, \Delta).\sigma_i \triangleq \delta \leq T.\sigma_{c_1(i)} - T.\sigma_{c_2(i)} \leq \Delta$$

avec $\delta \in \mathbb{Z} \cup \{-\infty\}$ *et* $\Delta \in \mathbb{Z} \cup \{+\infty\}$

Définition 94. *Lag. Pour une exécution* σ*, le prédicat de lag pour un état* σ_i *et pour deux chemins* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2 *est défini par :*

$$\mathtt{Lag}(P_1, c_1, P_2, c_2, \delta, \Delta).\sigma_i \triangleq \delta \leq \hat{x}_1.\sigma_{c_1(i)} - \hat{x}_2.\sigma_{c_2(i)} \leq \Delta$$

avec $\delta \in \mathbb{Z} \cup \{-\infty\}$ *et* $\Delta \in \mathbb{Z} \cup \{+\infty\}$

L'objectif d'une propriété de cohérence entre deux chemins est de spécifier que le décalage le long des deux chemins est comparable tout au long de l'exécution. `Shift` est défini par rapport au décalage logique et spécifie que les images utilisent des occurrences

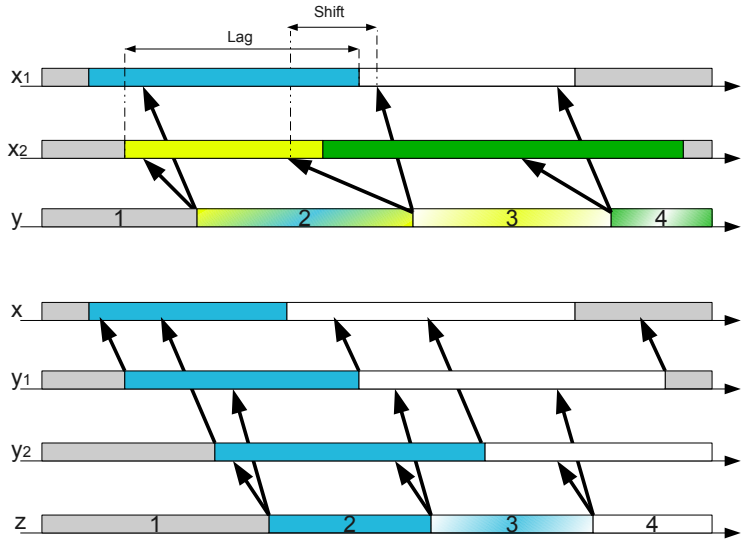

FIG. 28: Cohérence des chemins

des sources qui ont été prises dans des états proches. On considère donc la différence entres $T.\sigma_{c_1(i)}$ et $T.\sigma_{c_2(i)}$. Lag est défini par rapport aux événements de mise à jour et spécifie que les occurrences des deux sources sont apparues à des instants proches.

On illustre cette propriété avec deux exemples sur la figure 28. Les flèches vont de l'image à la source et pointent vers l'occurrence lue pour construire la nouvelle occurrence de l'image. Sur le premier exemple on a un calcul de y basé indirectement sur deux entrées x_1 et x_2. Il existe donc un ensemble de relations d'observations définissant deux chemins $[x_1, y]$ et $[x_2, y]$. Pour qu'une occurrence de y soit valide, on souhaite qu'elle soit basée sur des occurrences de x_1 et x_2 cohérentes. Le prédicat Lag est utilisé pour vérifier que les mises à jour des deux occurrences utilisées sont proches temporellement. Ainsi, pour l'occurrence 2 de y dans le premier cas de la figure 28, l'occurrence de y est liée à deux occurrences de x_1 et x_2 ayant commencé à des instants rapprochés. L'occurrence suivante notée 3 de y est basée sur une nouvelle occurrence de x_1 mais pas de x_2. Ces deux occurrences ont leurs débuts qui sont alors éloignés et la différence utilisée pour définir le prédicat de lag est grande. Ce prédicat de lag n'est pas respecté si les bornes données sont petites. Cependant, les états où ces deux occurrences étaient prises ne sont pas éloignées. La différence définissant le prédicat Shift est petite et un prédicat Shift avec des bornes raisonnables est satisfait avec deux horloges d'observation pointant vers des états rapprochés. Pour une telle configuration, si on met un prédicat de décalage Shift ayant comme borne 0 et 0, alors on veut que les occurrences utilisées aient été prises dans un même état. Le long des deux chemins, c'est alors comme si on avait eu une lecture synchrone des valeurs de x_1 et x_2.

Dans le deuxième exemple, on a trois observations $y_1 \preccurlyeq x, y_2 \preccurlyeq x, z \preccurlyeq f(y_1, y_2)$ qui définissent deux chemins $[x, y_1, z]$ et $[x, y_2, z]$ de x à z. On cherche à vérifier que les occurrences de x selon les deux chemins sont les mêmes c'est-à-dire que lors de la mise à jour de z avec les occurrences de y_1 et y_2, ces occurrences sont basées sur la même occurrence de x. L'occurrence de y notée occurrence 2, est bien construite avec la même occurrence de x selon les deux chemins. Cependant, l'occurrence suivante, notée 3, est construite avec deux occurrences différentes de x. Dans le cas d'une source commune

à deux chemins, en donnant des bornes nulles aux prédicats, on précise que l'on veut que ce soit la même occurrence qui soit vue le long des deux chemins. L'occurrence 3 ne serait alors pas valide.

En pratique et comme sur les exemples de la figure 28, ces propriétés sont utilisées pour deux chemins ayant des images communes. De plus, on pourrait étendre ces propriétés à un nombre supérieur de chemins, ce qui n'est pas fait ici.

On définit une forme normale pour ces propriétés.

Définition 95. *Forme normale d'une propriété sur un couple de chemin. Soit un couple de chemin* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2*. Une propriété* prop *sur ce couple de chemin est sous la forme normale si :*

$$\exists \delta_{Lag}, \delta_{Shi} \in \mathbb{Z} \cup \{-\infty\}, \exists \Delta_{Lag}, \Delta_{Shi} \in \mathbb{Z} \cup \{+\infty\} :$$
$$prop = P_1, P_2 \ \{Lag(\delta_{Lag}, \Delta_{Lag}), Shift(\delta_{Shi}, \Delta_{Shi})\}$$

Pour exprimer une forme normale équivalente, on utilise la proposition suivante :

Proposition 58. *Soit un couple de chemin* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2 *paramétré par un ensemble de prédicats* $\{Predicate_1(\delta_1, \Delta_1), \ldots, Predicate_n(\delta_n, \Delta_n)\}$*, alors on a :*

$$P_1, P_2 \ \{Predicate_1(\delta_1, \Delta_1), \ldots, Predicate_n(\delta_n, \Delta_n)\} \Leftrightarrow$$
$$P_1, P_2 \ \{Lag(\delta_{Lag}, \Delta_{Lag}), Shift(\delta_{Shi}, \Delta_{Shi})\}$$

où :

$$\delta_{Lag} = max(\{-\infty\} \cup \{\delta_i | Predicate_i = Lag\})$$
$$\Delta_{Lag} = min(\{+\infty\} \cup \{\Delta_i | Predicate_i = Lag\})$$
$$\delta_{Shi} = max(\{-\infty\} \cup \{\delta_i | Predicate_i = Shift\})$$
$$\Delta_{Shi} = min(\{+\infty\} \cup \{\Delta_i | Predicate_i = Shift\})$$

Démonstration. Toute différence est comprise entre $-\infty$ et $+\infty$. S'il n'y a pas de prédicat de lag ou de décalage, on utilise donc ces bornes dans la forme normale. Si ces prédicats sont déjà utilisés pour définir les propriétés d'un couple de chemin, on prend les bornes les plus strictes. D'après les propriétés d'un ensemble d'inégalités, on a donc une forme normale équivalente aux propriétés données sur un couple de chemins de propagation. □

Définition 96. *Paramètres des propriétés de cohérence sur les chemins. Pour une spécification* $\langle Archi, \mathcal{P}rop \rangle$*, on définit la fonction* PathParameters *pour un couple de chemins telle que :*

$$\forall P_1, P_2 \in \mathcal{P}aths(Archi) : PathParameters(P_1, P_2) \triangleq$$
$$\begin{cases} [\delta_{Lag}, \Delta_{Lag}, \delta_{Shi}, \Delta_{Shi}] \textit{ si} & P_1, P_2 \ \{Lag(\delta_{Lag1}, \Delta_{Lag1}), Shift(\delta_{Shi1}, \Delta_{Shi1})\} \in \mathcal{P}rop \\ [-\infty, +\infty, -\infty, +\infty] & \textit{sinon} \end{cases}$$

A.2 RÉGIME INITIAL D'UN COUPLE DE CHEMIN

Il est aussi possible que les propriétés de cohérence d'un couple de chemin ne soient pas vérifiées en l'état initial et dans les états postérieurs. De même que pour les propriétés sur un chemin, on définit le temps de sortie du régime initial d'un couple de chemins de propagation et on redéfinit les propriétés sur un couple de chemins.

Définition 97. *Temps maximum de sortie du régime initial d'un couple de chemins. Soit une architecture* $Archi$ *et un ensemble de propriétés temps réel* $\mathcal{P}rop$*. Pour tout chemins* P_1 *de* x_1

vers y_1*et* P_2 *de* x_2 *vers* y_2 *bien définis par* $\mathcal{A}rchi$*, le temps maximum de sortie du régime initial de ce couple* P_1, P_2 *est défini comme un entier* Δ_{P_1,P_2} *tel que pour toute exécution* σ *vérifiant la spécification on a :*

$$\forall i \in \mathbb{N} : \forall c_1 \in \mathcal{C}(P_1).\sigma, \forall c_2 \in \mathcal{C}(P_2).\sigma : T.\sigma_i \geq \Delta_{P_1,P_2} \Rightarrow (\hat{\hat{x}}_1.\sigma_{c_1(i)} \neq 0 \wedge \hat{\hat{x}}_2.\sigma_{c_2(i)} \neq 0))$$

Un couple de chemin sort donc du régime initial lorsque les deux chemins de ce couple sont sortis de leurs régimes initiaux.

PROPRIÉTÉS DE COHÉRENCE D'UN COUPLE DE CHEMINS On redéfinit chacun des prédicats utilisés pour définir une propriété de cohérence en tenant compte du régime initial.

Définition 98. *Lag. Pour une exécution* σ*, le prédicat de lag pour un état* σ_i *et pour deux chemins* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2 *est défini par :*

$$Lag(P_1, c_1, P_2, c_2, \delta, \Delta).\sigma_i \triangleq$$
$$\delta \leq \hat{\hat{x}}_1.\sigma_{c_1(i)} - \hat{\hat{x}}_2.\sigma_{c_2(i)} \leq \Delta \vee (T.\sigma_i < \Delta_{P_1,P_2} \wedge (\hat{\hat{x}}_1.\sigma_{c_1(i)} = 0 \vee \hat{\hat{x}}_2.\sigma_{c_2(i)} = 0))$$

Définition 99. *Décalage. Pour une exécution* σ*, le prédicat de décalage pour un état* σ_i *et pour deux chemins* P_1 *de* x_1 *vers* y_1 *et* P_2 *de* x_2 *vers* y_2 *est défini par :*

$$Shift(P_1, c_1, P_2, c_2, \delta, \Delta).\sigma_i \triangleq$$
$$\delta \leq T.\sigma_{c_1(i)} - T.\sigma_{c_2(i)} \leq \Delta \vee (T.\sigma_i < \Delta_{P_1,P_2} \wedge (\hat{\hat{x}}_1.\sigma_{c_1(i)} = 0 \vee \hat{\hat{x}}_2.\sigma_{c_2(i)} = 0))$$

CHOIX DU TEMPS DE SORTIE DU RÉGIME INITIAL D'UN COUPLE DE CHEMIN On définit tout simplement ce temps de sortie à l'aide de la proposition suivante :

Choix 7. *Soit une architecture* $\mathcal{A}rchi$ *et un ensemble de propriétés temps réel* $\mathcal{P}rop$*. Pour tout exécution* σ *vérifiant* $\mathcal{A}rchi$ *et* $\mathcal{P}rop$*, et pour tout chemin* P_1*,* P_2 *bien définis par* $\mathcal{A}rchi$ *et tel qu'il existe une propriété de cohérence sur ce couple :*

$$P_1, P_2 \{Lag(\delta_{Lag}, \Delta_{Lag}), Shift(\delta_{Shi}, \Delta_{Shi})\}$$

alors le temps maximum de sortie du régime initial de ce couple P_1, P_2 *est choisi égal à :*

$$\Delta_{P_1,P_2} = max(\Delta_{P_1}, \Delta_{P_2})$$

Justification. Soit deux chemins P_1 de x_2 vers $y_2$et P_2 de x_2 vers y_2. Si pour une exécution vérifiant $\mathcal{A}rchi$ et $\mathcal{P}rop$, on suppose que l'on est dans un état i tel que

$$\begin{array}{ll} & T.\sigma_i \geq max(\Delta_{P_1}, \Delta_{P_2}) \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & T.\sigma_i \geq \Delta_{P_1} \wedge T.\sigma_i \geq \Delta_{P_2} \\ \Rightarrow & \quad \{\text{temps maximum de sortie du régime initial d'un chemin}\} \\ & \forall c_1 \in \mathcal{C}(P_1).\sigma : \hat{\hat{x}}.\sigma_{c_1(i)} \neq 0 \wedge \forall c_2 \in \mathcal{C}(P_2).\sigma : \hat{\hat{x}}.\sigma_{c_2(i)} \neq 0 \\ \Rightarrow & \quad \{\text{réécriture}\} \\ & \forall c_1 \in \mathcal{C}(P_1).\sigma, \forall c_2 \in \mathcal{C}(P_2).\sigma : \hat{\hat{x}}.\sigma_{c_1(i)} \neq 0 \wedge \hat{\hat{x}}.\sigma_{c_2(i)} \neq 0 \end{array}$$

Le couple sort du régime initial lorsque les deux chemins sont sortis du régime initial.

On reformule la propriété de cohérence de deux chemins afin de pouvoir construire état par état une exécution qui vérifie cette propriété. On utilise alors deux propositions similaires à celles qui ont été données pour les propriétés d'un chemin de propagation.

Proposition 59. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ *et un couple de chemin* P_1 *et* P_2 *bien défini par* $\mathcal{A}rchi$ *respectivement d'une variable* x_1 *vers une variable* y *et d'une variable* x_2 *vers la même variable* y*. On démontre l'équivalence suivante :*

$$\exists c_1 \in \mathcal{C}(P_1).\sigma, \exists c_2 \in \mathcal{C}(P_2).\sigma, \forall i \in \mathbb{N} : \left(\begin{array}{c} Predicate_1(P_1, c_1, P_2, c_2, \delta_1, \Delta_1).\sigma_i \\ \wedge \ \ldots \ \wedge \\ Predicate_n(P_1, c_1, P_2, c_2, \delta_n, \Delta_n).\sigma_i \end{array} \right)$$
$$\Leftrightarrow$$
$$\forall\, i \in \mathbb{N}, \exists c_1 \in \mathcal{C}(P_1).\sigma, \exists c_2 \in \mathcal{C}(P_2).\sigma :$$
$$\hat{\hat{x}}_1.\sigma_{c_1(i)} \geq \hat{\hat{x}}_1.\sigma_{c_1(i-1)} \ \wedge \ \hat{\hat{x}}_2.\sigma_{c_2(i)} \geq \hat{\hat{x}}_2.\sigma_{c_2(i-1)} \ \wedge \left(\begin{array}{c} Predicate_1(P_1, c_1, P_2, c_2, \delta_1, \Delta_1).\sigma_i \\ \wedge \ \ldots \ \wedge \\ Predicate_n(P_1, c_1, P_2, c_2, \delta_n, \Delta_n).\sigma_i \end{array} \right)$$

et où

$$\forall\, i \in [1..n] : Predicate_i \in \{Latency, Lag, Shift\}$$

Justification. On a une preuve semblable à celle effectuée dans le cadre des propriétés d'un chemin de propagation. En prenant à chaque fois la plus petite valeur vérifiant les propriétés et quelque soit le comportement de l'image vis à vis des occurrences des sources, on prouve qu'on construit nécessairement des horloges croissantes pour les deux chemins.

Proposition 60. *Soit une spécification* $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$*, un chemin* P_1 *de la variable* x_1 *vers la variable* y *et un chemin* P_2 *de la variable* x_2 *vers la variable* y*, tous deux bien définis par* $\mathcal{A}rchi$*. On suppose que l'on a un ensemble de paramètres* S_Δ *tel que :*

$$\delta_{Shi}, \delta_{Lag} \in \mathbb{Z} \cup \{-\infty\} \ \wedge \ \Delta_{Shi}, \Delta_{Lag} \in \mathbb{N} \cup \{+\infty\}$$

Soit une exécution σ *vérifiant la spécification. On suppose que l'on a une occurrence de la source* x_1 *du chemin* P_1 *apparue à l'instant* upd_1 *et conservée pendant* dur_1 *et une occurrence de la source* x_2 *du chemin* P_2 *apparue à l'instant* upd_1 *et conservée pendant* dur_2*. Pour tout entier* i*, on a l'équivalence suivante :*

$$\begin{array}{l} \exists c_1 \in \mathcal{C}(\mathbb{N}), \exists c_2 \in \mathcal{C}(\mathbb{N}).\sigma : \\ (\hat{\hat{x}}.\sigma_{c_1(i)} = upd_1 \ \wedge \ \hat{\hat{x}}.\sigma_{c_2(i)} = upd_2) \wedge \\ Shift(P_1, c_1, P_2, c_2, \delta_{Shi}, \Delta_{Shi}).\sigma_i \ \wedge \ Lag(P_1, c_1, P_2, c_2, \delta_{Lag}, \Delta_{Lag}).\sigma_i \end{array}$$
$$\Leftrightarrow$$
$$upd_1 = 0 \ \vee \ upd_2 = 0 \ \vee \ Consistency(upd_1, dur_1, upd_2, dur_2, S_\Delta)$$

où l'on a Consistency *défini par la spécification égal à :*

$$\begin{array}{l} Consistency(upd_1, dur_1, upd_2, dur_2, S_\Delta) \triangleq \\ ]upd_1 - upd_2 - dur_2, upd_1 + dur_1 - upd_2[\cap [\delta_{Shi}, \Delta_{Shi}] \neq \varnothing \\ \wedge \ \delta_{Lag} \leq upd_1 - upd_2 \leq \Delta_{Lag} \end{array}$$

Démonstration. On suppose que l'on a une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$, un chemin P_1 de la variable x_1 vers la variable y et un chemin P_2 de la variable x_2 vers la variable y, tous deux bien définies par $\mathcal{A}rchi$. Soit une exécution vérifiant cette spécification et un entier i. On suppose que l'on a :

$$\begin{array}{l}\exists c_1 \in \mathcal{C}(\mathbb{N}), \exists c_2 \in \mathcal{C}(\mathbb{N}) : \\ \hat{\hat{x}}.\sigma_{c_1(i)} = upd_1 \ \wedge \ \hat{\hat{x}}.\sigma_{c_2(i)} = upd_2) \wedge \\ Shift(P_1, c_1, P_2, c_2, \delta_{Shi}, \Delta_{Shi}).\sigma_i \ \wedge \ Lag(P_1, c_1, P_2, c_2, \delta_{Lag}, \Delta_{Lag}).\sigma_i\end{array}$$

Dans cet état donc, si ce sont les horloges c_1 et c_2 qui caractérisent le décalage le long des chemins P_1 et P_2, alors les propriétés de cohérence sont vérifiées. Dans le cas où l'on a $upd_1 = 0 \vee upd_2 = 0$, on est bien dans le régime initial. On suppose que l'on a quitté ce régime initial et que donc $upd_1 \neq 0 \wedge upd_2 \neq 0$. Concernant le lag, on a une inégalité découlant donc directement de la définition. On s'intéresse au prédicat de décalage. On sait que :

$$\begin{array}{ll} & \forall c_1 \in \mathcal{C}(P_1).\sigma : \hat{\hat{x}}_1.\sigma_{c_1(i)} \leq T.\sigma_{c_1(i)} < \hat{\hat{x}}_1.\sigma_{c_1(i)} + d_{\bar{x_1}}.\sigma_{c_1(i)} \\ & \wedge \ \forall c_2 \in \mathcal{C}(P_2).\sigma : \hat{\hat{x}}_2.\sigma_{c_2(i)} \leq T.\sigma_{c_2(i)} < \hat{\hat{x}}_2.\sigma_{c_2(i)} + d_{\bar{x_2}}.\sigma_{c_2(i)} \\ \Leftrightarrow & \quad \{\text{multiplication par -1 et sachant que } \hat{\hat{x}}_2.\sigma_{c_2(i)} \geq 0\} \\ & \forall c_1 \in \mathcal{C}(P_1).\sigma : \hat{\hat{x}}_1.\sigma_{c_1(i)} \leq T.\sigma_{c_1(i)} < \hat{\hat{x}}_1.\sigma_{c_1(i)} + d_{\bar{x_1}}.\sigma_{c_1(i)} \\ & \wedge \ \forall c_2 \in \mathcal{C}(P_2).\sigma : -\hat{\hat{x}}_2.\sigma_{c_2(i)} - d_{\bar{x_2}}.\sigma_{c_2(i)} < -T.\sigma_{c_2(i)} \leq -\hat{\hat{x}}_2.\sigma_{c_2(i)} \\ \Rightarrow & \quad \{\text{ajout des inégalités}\} \\ & \forall c_1 \in \mathcal{C}(P_1).\sigma, \forall c_2 \in \mathcal{C}(P_2).\sigma : \\ & \hat{\hat{x}}_1.\sigma_{c_1(i)} - \hat{\hat{x}}_2.\sigma_{c_2(i)} - d_{\bar{x_2}}.\sigma_{c_2(i)} < T.\sigma_{c_1(i)} - T.\sigma_{c_2(i)} < \hat{\hat{x}}_1.\sigma_{c_1(i)} + d_{\bar{x_1}}.\sigma_{c_1(i)} - \hat{\hat{x}}_2.\sigma_{c_2(i)} \end{array}$$

Cet intervalle est continu et décrit toutes les valeurs qui peuvent être prises par $T.\sigma_{c_1(i)} - T.\sigma_{c_2(i)}$. On en déduit donc que pour que l'inégalité du décalage soit respectée, il faut qu'il y ait une intersection entre les deux intervalles et donc que :

$$]\hat{\hat{x}}_1.\sigma_{c_1(i)} - \hat{\hat{x}}_2.\sigma_{c_2(i)} - d_{\bar{x_2}}.\sigma_{c_2(i)}, \hat{\hat{x}}_1.\sigma_{c_1(i)} + d_{\bar{x_1}}.\sigma_{c_1(i)} - \hat{\hat{x}}_2.\sigma_{c_2(i)}[\cap [\delta_{Shi}, \Delta_{Shi}] \neq \varnothing$$

Inversement, si leur intersection n'est pas vide, alors il existe deux horloges c_1, c_2 respectivement de $\mathcal{C}(P_1).\sigma$ et $\mathcal{C}(P_2).\sigma$ tel que la différence $T.\sigma_{c_1(i)} - T.\sigma_{c_2(i)}$ soit comprise entre δ_{Shi} et Δ_{Shi}. □

On peut donc vérifier les propriété de cohérence en vérifiant la validité du prédicat Consistent. On donne ces propriétés pour un couple de chemin ayant la même image. Le prédicat Consistent doit donc être utilisé pour compléter la définition de l'action d'une variable dans les systèmes de transitions équivalent à une spécification.

Définition 100. *Soit une spécification $\langle \mathcal{A}rchi, \mathcal{P}rop \rangle$ et L_S la longueur de l'intervalle d'analyse de cette spécification, tel qu'il existe une observation $y \lesssim f(X)$ dans l'architecture du système. Pour que les propriétés de cohérence du système soit vérifiées, l'action de la variable y doit aussi vérifiée que l'on a :*

$$\begin{array}{l}\forall \langle upd_1, dur_1, P_1 \rangle \in min(H'_{y \lesssim f(X)}), \forall \langle upd_2, dur_2, P_2 \rangle \in min(H'_{y \lesssim f(X)}) : \\ \left(\begin{array}{l} upd_1 = 0 \ \vee \ upd_2 = 0 \vee \\ Consistency \left(\begin{array}{l} ValMod(upd_1, T', L_S), dur_1, ValMod(upd_2, T', L_S), \\ dur_2, PathParameters(P_1, P_2) \end{array} \right) \end{array} \right)\end{array}$$

B

AUTOMATISATION DE LA VÉRIFICATION DE LA SATISFIABILITÉ D'UNE SPÉCIFICATION

Il s'agit ici d'expliquer comment le système de transitions fini défini par une spécification est en pratique utilisé pour vérifier la satisfiabilité de cette spécification. Le but est d'utiliser un outil permettant donc d'automatiser la vérification de cette satisfiabilité.

B.1 VÉRIFICATION D'UN EXEMPLE PAR TLC

On utilise un outil de vérification et de model checking existant. On s'intéresse à des outils permettant de vérifier l'existence d'exécutions infinies définies par le système de transitions fini. La syntaxe utilisée pour donner la spécification du système de transitions fini étant proche de TLA, le choix de l'outil TLC (TLA model checker) est naturel. Il s'agit en effet d'un outil permettant la vérification de spécification écrite dans le langage TLA. On montre ici comment un exemple simple est vérifié en utilisant l'outil TLC

B.1.1 *Spécification de l'exemple*

Cet exemple est basé sur l'architecture suivante :

$$\mathcal{A}rchi \triangleq {}'x_1 \preccurlyeq x_1, {}'x_2 \preccurlyeq x_2, y \preccurlyeq f({}'x_1, {}'x_2)$$

Il y a donc trois observations et les occurrences des variables x_1 et x_2 sont propagées dans le système via les variables $'x_1$ et $'x_2$ pour mettre à jour la variable y. Les propriétés temporelles sont alors les suivantes :

$$\begin{aligned}
\mathcal{P}rop \triangleq\ & x_1\{Sporadic(1,1)\} \\
& x_2\{Sporadic(2,2)\} \\
& y\{Sporadic(2,2)\} \\
& [x_1, {}'x_1]\{Lag(1,+\infty\} \\
& [x_2, {}'x_2]\{Lag(1,+\infty\} \\
& [{}'x_1, y]\{Lag(1,+\infty\} \\
& [{}'x_2, y]\{Lag(1,+\infty\} \\
& [x_1, {}'x_1, y]\{Shift(0,3)\} \\
& [x_2, {}'x_2, y]\{Shift(0,3)\}
\end{aligned}$$

Les trois variables x_1, x_2 et y sont sporadiques. x_1 est cependant mis à jour plus rapidement que les deux autres variables. Pour chaque chemin correspondant à une observation, on a une propriété de lag avec une borne inférieure. Ce minimum impose qu'une unité de temps se soit écoulée avant qu'une nouvelle occurrence d'une source d'une observation soit utilisée pour définir une nouvelle occurrence de l'image. Il y a de plus deux propriétés de décalage entre x_1 et y et entre x_2 et y. Ces propriétés ont une borne supérieure limitant le décalage le long des chemins de propagation.

B.1.2 *Spécification en TLA*

Les actions de chaque variables sont données en TLA. Les variables historiques sont implantées par des séquences au lieu d'être des tableaux. Chaque ensemble de caractéristiques d'occurrences est représenté par un tableau, chaque élément du tableau est lui même un tableau donnant les caractéristiques d'une occurrence c'est à dire la date de mise à jour et la durée de l'occurrence.

Les actions de chaque variable sont simplifiées par rapport aux définitions données dans le chapitre 7.2.2 page 146. Seuls les prédicats correspondant aux propriétés de la spécification qui ne sont pas nécessairement vérifiées sont spécifiés.

La spécification donnée ici en TLA inclut un régime initial pour la variable y qui doit attendre la propagation des occurrences des entrées le long des chemins pour pouvoir être mise à jour. La longueur de la période d'analyse est calculée en fonction de la sporadicité des entrées du système et du décalage le long des chemins selon le graphe de calcul du temps de sortie du régime initial introduit par la définition 60 page 94.

On spécifie une propriété d'interblocage $\Diamond \Box (T' = T)$. Cette propriété est vraie lorsque pour une exécution, le temps finit par ne plus évoluer. Il s'agit donc bien d'une propriété d'interblocage, les autres variables ne pouvant qu'évoluer en même temps que la variable T. L'outil TLC cherche si il existe des exécutions ne vérifiant pas cette propriété. Il s'agit donc d'exécutions infinies. L'existence d'une telle exécution prouve alors la satisfiabilité de la spécification.

MODULE *exemple*3

EXTENDS *Naturals*, *Sequences*, *TLC*

VARIABLES

T,	le temps
y, dy,	$\hat{y}, d_{\bar{y}}$
$x1$, $dx1$,	$\hat{x}_1, d_{\bar{x}_1}$
$x2$, $dx2$,	$\hat{x}_2, d_{\bar{x}_2}$
$px1$, $dpx1$,	$`\hat{x}_1, d_{`\bar{x}_1}$
$px2$, $dpx2$,	$`\hat{x}_2, d_{`\bar{x}_2}$
$Hx1$,	$H_{`x_1 \preceq x_1}$
$Hx2$,	$H_{`x_2 \preceq x_2}$
Hy	$H_{y \preceq f(x_1,x_2)}$

$Period \triangleq 7$

$AnalysisInterval \triangleq 0 \,..\, 13$

$CoupleDateDuree \triangleq 1 \,..\, 2$

$QuadrupletX1X2Px1Px2 \triangleq 1 \,..\, 4$

$ValMod(u,\, t) \triangleq t - ((((t - u)\%Period) + Period)\%Period)$

$AllVars \triangleq \langle T,\, y,\, dy,\, x1,\, dx1,\, x2,\, dx2,\, px1,\, dpx1,\, px2,\, dpx2,\, Hx1,\, Hx2,\, Hy \rangle$

$Typage \triangleq$
$\land\ T \in AnalysisInterval$
$\land\ y \in AnalysisInterval \land dy \in AnalysisInterval$
$\land\ x1 \in AnalysisInterval \land dx1 \in AnalysisInterval$
$\land\ x2 \in AnalysisInterval \land dx2 \in AnalysisInterval$
$\land\ px1 \in AnalysisInterval \land dpx1 \in AnalysisInterval$
$\land\ px2 \in AnalysisInterval \land dpx2 \in AnalysisInterval$
$\land\ Hx1 \in Seq([CoupleDateDuree \to AnalysisInterval])$
$\land\ Hx2 \in Seq([CoupleDateDuree \to AnalysisInterval])$
$\land\ Hy \in Seq([QuadrupletX1X2Px1Px2 \to [CoupleDateDuree \to AnalysisInterval]])$

Si le *syst*me est faisable (= *il* existe une excution infinie), cette propri*t* est invalide. Echec de la vrification par *TLC* = *syst*me faisable.

$Interblocage \triangleq \Diamond\Box(T' = T)$

L'ensemble des suffixes de s, $i.e\{s, tail(s), tail(tail(s)), \ldots tail\hat{}n(s), \langle\rangle\}$.

$Suffixes(s) \triangleq$
$\{SubSeq(s, i, Len(s)) : i \in 1 \ldots Len(s) + 1\}$

$Init \triangleq$
$\land T = 0$
$\land y = 0 \land dy = 1$
$\land x1 = 0 \land dx1 = 1$
$\land x2 = 0 \land dx2 = 1$
$\land px1 = 0 \land dpx1 = 1$
$\land px2 = 0 \land dpx2 = 1$
$\land Hx1 = \langle\langle x1, dx1\rangle\rangle$
$\land Hx2 = \langle\langle x2, dx2\rangle\rangle$
$\land Hy = \langle\langle\langle x1, dx1\rangle, \langle x2, dx2\rangle, \langle px1, dpx1\rangle, \langle px2, dpx2\rangle\rangle\rangle$

$ActionT \triangleq T' = \text{IF } (T + 1) = 2 * Period \text{ THEN } (T + 1) - Period \text{ ELSE } (T + 1)$

$ActionX1 \triangleq$
$\land \lor T + 1 - ValMod(x1, T) < 1 \land x1' = x1$
$\lor T + 1 - ValMod(x1, T) \geq 1 \land x1' = T'$
$\land dx1' = T' + 1 - ValMod(x1', T')$

$ActionX2 \triangleq$
$\land \lor T + 1 - ValMod(x2, T) < 2 \land x2' = x2$
$\lor T + 1 - ValMod(x2, T) \geq 2 \land x2' = T'$
$\land dx2' = T' + 1 - ValMod(x2', T')$

$ActionPx1 \triangleq$
$\land \exists f \in Suffixes(Hx1) :$
$\land Hx1' = Append(f, \langle x1', dx1'\rangle)$
$\land \lor \land px1' = px1$
$\land Head(Hx1')[1] = Head(Hx1)[1]$
$\lor \land px1' = T'$
$\land Head(Hx1')[1] \neq Head(Hx1)[1]$
$\land T' - ValMod(Head(Hx1')[1], T') \geq 1$
$\land dpx1' = T' + 1 - ValMod(px1', T')$

$ActionPx2 \triangleq$
$\land \exists f \in Suffixes(Hx2) :$
$\land Hx2' = Append(f, \langle x2', dx2'\rangle)$
$\land \lor \land px2' = px2$
$\land Head(Hx2')[1] = Head(Hx2)[1]$
$\lor \land px2' = T'$

$$
\begin{array}{l}
\quad\quad\quad \land\ Head(Hx2')[1] \neq Head(Hx2)[1] \\
\quad\quad\quad \land\ T' - ValMod(Head(Hx2')[1],\ T') \geq 1 \\
\quad\quad \land\ dpx2' = T' + 1 - ValMod(px2',\ T') \\
\\
ActionY\ \triangleq \\
\quad \land\ \exists\, f \in Suffixes(Hy) : \\
\quad\quad \land\ Hy' = Append(f,\ \langle Head(Hx1'),\ Head(Hx2'),\ \langle px1',\ dpx1' \rangle,\ \langle px2',\ dpx2' \rangle \rangle) \\
\quad\quad \land\ \lor\ \land\ y' = y \land (Head(Hy')[3][1] = Head(Hy)[3][1] \land Head(Hy')[4][1] = Head(Hy)[4][1]) \\
\quad\quad\quad\quad \land\ \lor\ (T' < 5 \land y = 0) \\
\quad\quad\quad\quad\quad \lor\ \land\ T + 1 - ValMod(y,\ T) < 1 \\
\quad\quad\quad\quad\quad\quad \land\ Head(Hy')[1][2] \geq T' - ValMod(Head(Hy')[1][1],\ T') - 1 \\
\quad\quad\quad\quad\quad\quad \land\ Head(Hy')[2][2] \geq T' - ValMod(Head(Hy')[2][1],\ T') - 1 \\
\quad\quad\quad \lor\ \land\ y' = T' \\
\quad\quad\quad\quad \land\ T + 1 - ValMod(y,\ T) \geq 1 \\
\quad\quad\quad\quad \land\ (Head(Hy')[3][1] \neq Head(Hy)[3][1] \lor Head(Hy')[4][1] \neq Head(Hy)[4][1]) \\
\quad\quad\quad\quad \land\ T' - ValMod(Head(Hy')[3][1],\ T') \geq 1 \\
\quad\quad\quad\quad \land\ T' - ValMod(Head(Hy')[4][1],\ T') \geq 1 \\
\quad\quad\quad\quad \land\ Head(Hy')[1][2] \geq T' - ValMod(Head(Hy')[1][1],\ T') - 1 \\
\quad\quad\quad\quad \land\ Head(Hy')[2][2] \geq T' - ValMod(Head(Hy')[2][1],\ T') - 1 \\
\quad\quad \land\ dy' = T' + 1 - ValMod(y',\ T') \\
\\
Debug\ \triangleq\ PrintT(AllVars) \\
\\
Next\ \triangleq\ \land\ ActionT \\
\quad\quad\quad\quad \land\ ActionX1 \\
\quad\quad\quad\quad \land\ ActionX2 \\
\quad\quad\quad\quad \land\ ActionPx1 \\
\quad\quad\quad\quad \land\ ActionPx2 \\
\quad\quad\quad\quad \land\ ActionY \\
\\
Spec\ \triangleq\ Init \land \Box[Next]_{AllVars}
\end{array}
$$

B.1.3 *Résultats*

La vérification effectuée par TLC donne le résultat suivant.

```
TLC Version 2.01 of April 9, 2008
Model-checking
Parsing file exemple3.tla
Parsing file .../TLA/tla/tlasany/StandardModules/Naturals.tla
Parsing file .../TLA/tla/tlasany/StandardModules/Sequences.tla
Parsing file .../TLA/tla/tlasany/StandardModules/TLC.tla
Semantic processing of module Naturals
Semantic processing of module Sequences
Semantic processing of module TLC
Semantic processing of module exemple3
Implied-temporal checking--satisfiability problem has 1 branches.
Finished computing initial states: 1 distinct state generated.
--Checking temporal properties for the complete state space...
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
    calculated (optimistic):  1.638416507500351E-13
    based on the actual fingerprints:  6.359399434140213E-14
4106 states generated, 961 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 6.
```

Ici, aucune erreur n'a été trouvée. Il n'existe donc pas d'exécution vérifiant la propriété d'interblocage. On en déduit donc que la spécification n'est pas satisfiable. Ceci est du au fait que la propriété de décalage $[x_1, 'x_1, y]\{Shift(0,3)\}$ n'est pas vérifiée. En effet, x_1 est mise à jour à chaque unité de temps alors que y est mise à jour toutes les deux unités de temps. Seulement une occurrence de x_1 sur deux sont donc utilisées pour mettre à jour y. Du fait du décalage introduit le long du chemin $[x_1, 'x_1, y]$ par les propriétés de lag, lorsqu'une occurrence de x_1 est utilisée pour définir une occurrence de y, celle ci a été remplacée par une nouvelle occurrence de x_1 il y a trop longtemps pour que la propriété de décalage puisse être respectée.

Une modification peut être apportée pour que la spécification soit satisfiable. On modifie les propriétés temporelles du système pour que la variable x_1 ait les mêmes caractéristiques de sporadicité que x_2 et soit donc mise à jour moins rapidement. Alors chaque occurrence de y est basée sur une occurrence de x_1 qui était conservée dans des instants suffisamment récents pour que la propriété de décalage soit vérifiée. On a alors les propriétés temporelles suivantes :

$$
\begin{aligned}
\mathcal{P}rop \triangleq\ & x_1\{Sporadic(2,2)\} \\
& x_2\{Sporadic(2,2)\} \\
& y\{Sporadic(2,2)\} \\
& [x_1, 'x_1]\{Lag(1,+\infty\} \\
& [x_2, 'x_2]\{Lag(1,+\infty\} \\
& ['x_1, y]\{Lag(1,+\infty\} \\
& ['x_2, y]\{Lag(1,+\infty\} \\
& [x_1, 'x_1, y]\{Shift(0,3)\} \\
& [x_2, 'x_2, y]\{Shift(0,3)\}
\end{aligned}
$$

En modifiant en conséquences les sources TLA, TLC indique alors comme résultat que la propriété d'interblocage n'est pas vérifiée. La trace d'une exécution ne vérifiant pas

cette propriété est donnée. Cette trace correspond à une exécution infinie prouvant la satisfiabilité de la spécification.

```
TLC Version 2.01 of April 9, 2008
Model-checking
Parsing file exemple3.tla
Parsing file .../TLA/tla/tlasany/StandardModules/Naturals.tla
Parsing file .../TLA/tla/tlasany/StandardModules/Sequences.tla
Parsing file .../TLA/tla/tlasany/StandardModules/TLC.tla
Semantic processing of module Naturals
Semantic processing of module Sequences
Semantic processing of module TLC
Semantic processing of module exemple3
Implied-temporal checking--satisfiability problem has 1 branches.
Finished computing initial states: 1 distinct state generated.
--Checking temporal properties for the complete state space...
Error: Temporal properties were violated.
The following behaviour constitutes a counter-example:

STATE 1: <Initial predicate>
/\ Hx2 = <<<<0, 1>>>>
/\ px1 = 0
/\ px2 = 0
/\ Hy = <<<<<<0, 1>>, <<0, 1>>, <<0, 1>>, <<0, 1>>>>>>
/\ T = 0
/\ y = 0
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 0
/\ x2 = 0
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<0, 1>>>>

STATE 2: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<0, 2>>>>
/\ px1 = 0
/\ px2 = 0
/\ Hy = <<<<<<0, 2>>, <<0, 2>>, <<0, 2>>, <<0, 2>>>>>>
/\ T = 1
/\ y = 0
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 0
/\ x2 = 0
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<0, 2>>>>

STATE 3: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<0, 2>>, <<2, 1>>>>
/\ px1 = 0
```

```
/\ px2 = 0
/\ Hy = <<<<<<0, 2>>, <<0, 2>>, <<0, 3>>, <<0, 3>>>>>>
/\ T = 2
/\ y = 0
/\ dpx1 = 3
/\ dpx2 = 3
/\ x1 = 2
/\ x2 = 2
/\ dy = 3
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<0, 2>>, <<2, 1>>>>

STATE 4: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<2, 2>>>>
/\ px1 = 3
/\ px2 = 3
/\ Hy = << <<<<0, 2>>, <<0, 2>>, <<0, 3>>, <<0, 3>>>>,
   <<<<2, 2>>, <<2, 2>>, <<3, 1>>, <<3, 1>>>> >>
/\ T = 3
/\ y = 0
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 2
/\ x2 = 2
/\ dy = 4
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<2, 2>>>>

STATE 5: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<2, 2>>, <<4, 1>>>>
/\ px1 = 3
/\ px2 = 3
/\ Hy = <<<<<<2, 2>>, <<2, 2>>, <<3, 2>>, <<3, 2>>>>>>
/\ T = 4
/\ y = 4
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 4
/\ x2 = 4
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<2, 2>>, <<4, 1>>>>

STATE 6: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<4, 2>>>>
/\ px1 = 5
/\ px2 = 5
/\ Hy = << <<<<2, 2>>, <<2, 2>>, <<3, 2>>, <<3, 2>>>>,
   <<<<4, 2>>, <<4, 2>>, <<5, 1>>, <<5, 1>>>> >>
/\ T = 5
/\ y = 4
```

```
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 4
/\ x2 = 4
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<4, 2>>>>

STATE 7: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<4, 2>>, <<6, 1>>>>
/\ px1 = 5
/\ px2 = 5
/\ Hy = <<<<<<4, 2>>, <<4, 2>>, <<5, 2>>, <<5, 2>>>>>>
/\ T = 6
/\ y = 6
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 6
/\ x2 = 6
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<4, 2>>, <<6, 1>>>>

STATE 8: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<6, 2>>>>
/\ px1 = 7
/\ px2 = 7
/\ Hy = << <<<<4, 2>>, <<4, 2>>, <<5, 2>>, <<5, 2>>>>,
   <<<<6, 2>>, <<6, 2>>, <<7, 1>>, <<7, 1>>>> >>
/\ T = 7
/\ y = 6
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 6
/\ x2 = 6
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<6, 2>>>>

STATE 9: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<6, 2>>, <<8, 1>>>>
/\ px1 = 7
/\ px2 = 7
/\ Hy = <<<<<<6, 2>>, <<6, 2>>, <<7, 2>>, <<7, 2>>>>>>
/\ T = 8
/\ y = 8
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 8
/\ x2 = 8
/\ dy = 1
```

```
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<6, 2>>, <<8, 1>>>>

STATE 10: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<8, 1>>, <<8, 2>>>>
/\ px1 = 9
/\ px2 = 9
/\ Hy = << <<<<6, 2>>, <<6, 2>>, <<7, 2>>, <<7, 2>>>>,
   <<<<8, 1>>, <<8, 1>>, <<9, 1>>, <<9, 1>>>> >>
/\ T = 9
/\ y = 8
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 8
/\ x2 = 8
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<8, 1>>, <<8, 2>>>>

STATE 11: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<8, 2>>, <<10, 1>>>>
/\ px1 = 9
/\ px2 = 9
/\ Hy = << <<<<8, 1>>, <<8, 1>>, <<9, 1>>, <<9, 1>>>>,
   <<<<8, 2>>, <<8, 2>>, <<9, 2>>, <<9, 2>>>> >>
/\ T = 10
/\ y = 10
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 10
/\ x2 = 10
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<8, 2>>, <<10, 1>>>>

STATE 12: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<10, 2>>>>
/\ px1 = 11
/\ px2 = 11
/\ Hy = << <<<<8, 2>>, <<8, 2>>, <<9, 2>>, <<9, 2>>>>,
   <<<<10, 2>>, <<10, 2>>, <<11, 1>>, <<11, 1>>>> >>
/\ T = 11
/\ y = 10
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 10
/\ x2 = 10
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<10, 2>>>>
```

```
STATE 13: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<10, 2>>, <<12, 1>>>>
/\ px1 = 11
/\ px2 = 11
/\ Hy = <<<<<<10, 2>>, <<10, 2>>, <<11, 2>>, <<11, 2>>>>>>
/\ T = 12
/\ y = 12
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 12
/\ x2 = 12
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<10, 2>>, <<12, 1>>>>

STATE 14: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<12, 2>>>>
/\ px1 = 13
/\ px2 = 13
/\ Hy = << <<<<10, 2>>, <<10, 2>>, <<11, 2>>, <<11, 2>>>>,
   <<<<12, 2>>, <<12, 2>>, <<13, 1>>, <<13, 1>>>> >>
/\ T = 13
/\ y = 12
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 12
/\ x2 = 12
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<12, 2>>>>

STATE 15: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<12, 2>>, <<7, 1>>>>
/\ px1 = 13
/\ px2 = 13
/\ Hy = <<<<<<12, 2>>, <<12, 2>>, <<13, 2>>, <<13, 2>>>>>>
/\ T = 7
/\ y = 7
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 7
/\ x2 = 7
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<12, 2>>, <<7, 1>>>>

STATE 16: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<7, 2>>>>
/\ px1 = 8
/\ px2 = 8
```

```
/\ Hy = << <<<<12, 2>>, <<12, 2>>, <<13, 2>>, <<13, 2>>>>,
   <<<<7, 2>>, <<7, 2>>, <<8, 1>>, <<8, 1>>>> >>
/\ T = 8
/\ y = 7
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 7
/\ x2 = 7
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<7, 2>>>>

STATE 17: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<7, 2>>, <<9, 1>>>>
/\ px1 = 8
/\ px2 = 8
/\ Hy = <<<<<<7, 2>>, <<7, 2>>, <<8, 2>>, <<8, 2>>>>>>
/\ T = 9
/\ y = 9
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 9
/\ x2 = 9
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<7, 2>>, <<9, 1>>>>

STATE 18: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<9, 2>>>>
/\ px1 = 10
/\ px2 = 10
/\ Hy = << <<<<7, 2>>, <<7, 2>>, <<8, 2>>, <<8, 2>>>>,
   <<<<9, 2>>, <<9, 2>>, <<10, 1>>, <<10, 1>>>> >>
/\ T = 10
/\ y = 9
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 9
/\ x2 = 9
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<9, 2>>>>

STATE 19: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<9, 2>>, <<11, 1>>>>
/\ px1 = 10
/\ px2 = 10
/\ Hy = <<<<<<9, 2>>, <<9, 2>>, <<10, 2>>, <<10, 2>>>>>>
/\ T = 11
/\ y = 11
/\ dpx1 = 2
```

```
/\ dpx2 = 2
/\ x1 = 11
/\ x2 = 11
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<9, 2>>, <<11, 1>>>>

STATE 20: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<11, 2>>>>
/\ px1 = 12
/\ px2 = 12
/\ Hy = << <<<<9, 2>>, <<9, 2>>, <<10, 2>>, <<10, 2>>>>,
   <<<<11, 2>>, <<11, 2>>, <<12, 1>>, <<12, 1>>>> >>
/\ T = 12
/\ y = 11
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 11
/\ x2 = 11
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<11, 2>>>>

STATE 21: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<11, 2>>, <<13, 1>>>>
/\ px1 = 12
/\ px2 = 12
/\ Hy = <<<<<<11, 2>>, <<11, 2>>, <<12, 2>>, <<12, 2>>>>>>
/\ T = 13
/\ y = 13
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 13
/\ x2 = 13
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<11, 2>>, <<13, 1>>>>

STATE 22: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<13, 2>>>>
/\ px1 = 7
/\ px2 = 7
/\ Hy = << <<<<11, 2>>, <<11, 2>>, <<12, 2>>, <<12, 2>>>>,
   <<<<13, 2>>, <<13, 2>>, <<7, 1>>, <<7, 1>>>> >>
/\ T = 7
/\ y = 13
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 13
/\ x2 = 13
/\ dy = 2
```

```
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<13, 2>>>>

STATE 23: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<13, 2>>, <<8, 1>>>>
/\ px1 = 7
/\ px2 = 7
/\ Hy = <<<<<<13, 2>>, <<13, 2>>, <<7, 2>>, <<7, 2>>>>>>
/\ T = 8
/\ y = 8
/\ dpx1 = 2
/\ dpx2 = 2
/\ x1 = 8
/\ x2 = 8
/\ dy = 1
/\ dx1 = 1
/\ dx2 = 1
/\ Hx1 = <<<<13, 2>>, <<8, 1>>>>

STATE 24: <Action line 116, col 9 to line 121, col 20 of module exemple3>
/\ Hx2 = <<<<8, 1>>, <<8, 2>>>>
/\ px1 = 9
/\ px2 = 9
/\ Hy = << <<<<13, 2>>, <<13, 2>>, <<7, 2>>, <<7, 2>>>>,
   <<<<8, 1>>, <<8, 1>>, <<9, 1>>, <<9, 1>>>> >>
/\ T = 9
/\ y = 8
/\ dpx1 = 1
/\ dpx2 = 1
/\ x1 = 8
/\ x2 = 8
/\ dy = 2
/\ dx1 = 2
/\ dx2 = 2
/\ Hx1 = <<<<8, 1>>, <<8, 2>>>>

STATE 25: <Action line 116, col 9 to line 121, col 20 of module exemple3>
Back to state 11.

12498 states generated, 1930 distinct states found, 0 states left on queue.
```

Une autre possibilité pour que la spécification soit vérifiable est de mettre plus rapidement à jour la variable y. On a alors les propriétés temporelles suivantes :

$$\begin{aligned}\mathcal{P}\text{rop} \triangleq\ & x_1\{\text{Sporadic}(1,1)\}\\ & x_2\{\text{Sporadic}(2,2)\}\\ & y\{\text{Sporadic}(1,1)\}\\ & [x_1, `x_1]\{\text{Lag}(1,+\infty\}\\ & [x_2, `x_2]\{\text{Lag}(1,+\infty\}\\ & [`x_1, y]\{\text{Lag}(1,+\infty\}\\ & [`x_2, y]\{\text{Lag}(1,+\infty\}\\ & [x_1, `x_1, y]\{\text{Shift}(0,3)\}\\ & [x_2, `x_2, y]\{\text{Shift}(0,3)\}\end{aligned}$$

B.1.4 *Limite à l'utilisation de TLC*

La principale limite lors de l'utilisation de TLC est que les performances de TLC décroissent rapidement quand le nombre d'états à explorer augmente. Pour des spécifications où le nombre de variables, le nombre de relations d'observation et la période d'analyse sont plus élevés, TLC ne peut alors pas vérifier la satisfiabilité.

Afin de s'assurer de la correction des résultats donnés par l'outil TLC, il faut tout d'abord s'assurer que l'implantation du système de transitions fini défini par une spécification en TLA est correcte. La traduction est directe et il est donc relativement aisé de s'assurer de sa correction. Cependant, afin de faciliter le travail de l'outil, il faut simplifier l'implantation en TLA. Ainsi et comme dans l'exemple, il ne faut pas implanter les éléments relatifs aux propriétés de la spécification qui sont systématiquement vérifiées. Cette simplification du système de transition complexifie alors la traduction en TLA.

Finalement, la validité du résultat donnée par l'outil TLC dépend de la fiabilité de cet outil. A l'usage, nous avons pu constater que cette fiabilité n'est pas exemplaire. Il faut pouvoir déterminer si dans le cas des modèles TLA définie à partir d'une spécification, cette fiabilité est suffisante.

B.2 VÉRIFICATION D'UN EXEMPLE PAR SPIN

B.2.1 *Présentation de Spin*

Un deuxième outil auquel on s'intéresse est l'outil Spin [Hol03] permettant de vérifier des spécifications formelles de système données dans le langage Promela. Ce langage est inspiré de l'algèbre de processus CSP et généralement utilisé pour la spécification de systèmes distribués et l'étude de la concurrence. La sémantique du langage Promela ne permet pas une traduction directe de la définition d'un système de transitions fini défini par une spécification dans ce langage. Une difficulté est donc d'assurer cette traduction et de prouver sa validité.

B.2.2 *Spécification en Promela*

En Promela, on implémente les actions de chaque variable à l'aide de macro. Chaque macro définit donc si la variable est mise à jour ou non et éventuellement avec quelles occurrences des autres variables. Ces différentes actions sont ensuite exécutées séquentiellement, en commençant par exécuter les macros des entrées du systèmes et en suivant la propagation des occurrences le long de ces chemins.

On définit les variables historiques grâce au canaux de communications de Promela. Ces canaux sont des files FIFO permettant le passage de messages entre les processus. On s'en sert donc ici pour communiquer les ensembles de caractéristiques des occurrences source à l'image.

Finalement, afin de vérifier la satisfiabilité de la spécification en utilisant l'outil Spin, on recherche les "acceptance cycle". Il s'agit d'ajouter un label spécifiant un état comme un état acceptant. L'outil Spin fait alors une recherche d'exécutions passant infiniment par un tel état considèrant de tels exécutions comme des erreurs. L'état acceptant est ici l'état précédant la décrémentation de la variable T lors de la fin d'une période d'analyse. En prouvant l'existence d'exécutions passant infiniment souvent par cet état, Spin prouve donc la satisfiabilité d'une spécification.

La spécification en Promela est donc la suivante :

```
/* La longueur de l'intervalle d'analyse */
#define P 7

/* Calcul de la différence entre deux variables en tenant compte du modulo */
#define DifMod(t, x)  (((t - x) %P + P) %P)

/* Les dates de mises à jour des variables du système */
int x_1, x_2, px_1, px_2, y;
/* Les durées de mises à jour */
int d_x_1, d_x_2;
/* Les variables historiques */
chan H_11 = [P] of {int, int};
chan H_22 = [P] of {int, int};
chan H_y = [P] of {int, int, int, int, int, int}

/* Les variables en tête de chaque historique
 * Ce sont les dates d'apparition des occurrences utilisées pour
 * définir l'occurrence de l'image en chaque état.
 */
int head_x_1, head_x_2, head_x_1_2, head_x_2_2, head_px_1, head_px_2;

/* Le temps */
int T;

/* Gestion du temps */
inline Temps(T) {
   if
      :: (T + 1 == 2*P) -> accept: T = P;
      :: (T + 1 < 2*P) -> T = T + 1;
   fi
}

/* Action de la variable x_1 */

inline X1(T, x_1, d_x_1, H_11) {
   do
      /* Non mise à jour */
      :: DifMod(T, x_1) < 1 -> break;
      /* Mise à jour*/
      :: DifMod(T, x_1) >= 1 -> x_1 = T; break;
      od;
      /* Mise à jour de la durée de l'occurrence */
      d_x_1 = DifMod(T+1, x_1);
      /* Ajout de l'occurrence à l'historique
       * de l'observation entre x_1 et px_1
       */
      H_11! x_1, d_x_1;
}

/* Action de la variable x_2 */
inline X2(T, x_2, d_x_2, H_22) {
   do
      /* Non mise à jour */
      :: DifMod(T, x_2) < 2 -> break;
```

```
      /* Mise à jour*/
      :: DifMod(T, x_2) >= 2 -> x_2 = T; break;
   od;
   d_x_2 = DifMod(T+1, x_2);
   H_22! x_2, d_x_2;
}

/* Action de la variable px_1 */
inline X11(T, px_1, H_11, head_x_1, head_d_x_1) {
   do
      /* Non mise à jour
      * px_1 reste défini sur la même occurrence de x_1 */
      :: last_x_1 == head_x_1 -> break;
      /* Mise à jour
       * px_1 est défini sur une nouvelle occurrence qui vérifie le lag */
      :: (last_x_1 != head_x_1 && DifMod(T, head_x_1) >=1) -> px_1 = T; break;
      /* dépile une occurrence de x_1 qui est dans l'historique */
      :: H_11? head_x_1, head_d_x_1;
   od;
   last_x_1 = head_x_1;
}

/* Action de la variable px_2 */
inline X22(T, px_2, H_22, head_x_2, head_d_x_2) {
   do
      /* Non mise à jour
       * px_2 reste défini sur la même occurrence de x_2 */
      :: last_x_2 == head_x_2 -> break;
      /* Mise à jour
       * px_2 est défini sur une nouvelle occurrence qui vérifie le lag */
      :: (last_x_2 != head_x_2 && DifMod(T, head_x_2) >=1) -> px_2 = T; break;
      /* dépile une occurrence de x_2 qui est dans l'historique */
      :: H_22? head_x_2, head_d_x_2;
   od;
   last_x_2 = head_x_2;
}

/* Action de la variable y */
inline Y(T, y, H_y, head_x_1, head_x_2, head_d_x_1, head_d_x_2,
         px_1, px_2, head_x_1_2, head_x_2_2, head_px_1, head_px_2,
         head_d_x_1_2, head_d_x_2_2) {

   H_y! head_x_1, head_d_x_1, head_x_2, head_d_x_2, px_1, px_2;
   do
      /* Non mise à jour
       * y reste défini sur les mêmes occurrences des sources
       * soit on est dans le régime initial
       * soit les propriété de la spécification sont vérifiées
       */
      :: (T<5 && y == 0) ||
            (DifMod(T, y) < 2
            && (last_px_2 == head_px_2) && (last_px_1 == head_px_1)
            && (head_d_x_1_2 >=  DifMod(T, head_x_1_2) - 1)
            && (head_d_x_2_2 >=  DifMod(T, head_x_2_2) - 1))
```

```
          -> break;
      /* Mise à jour
       * y est défini sur de nouvelles occurrences des sources
       * les propriétés de la spécification doivent être vérifiées
       */
      :: (DifMod(T, y) >= 2)
         && ((last_px_2 != head_px_2) || (last_px_1 != head_px_1))
         && (DifMod(T, head_px_1) >= 1)
         && (DifMod(T, head_px_2) >= 1)
         && (head_d_x_1_2 >=  DifMod(T, head_x_1_2) - 1)
         && (head_d_x_2_2 >=  DifMod(T, head_x_2_2) - 1)
         -> y = T; break;
      /* dépile un ensemble de caractéristiques d'occurrence de l'historique */
      :: H_y? head_x_1_2, head_d_x_1_2, head_x_2_2, head_d_x_2_2, head_px_1, head_px_2;
   od;
   last_px_2 = head_px_2;
   last_px_1 = head_px_1;
}

/* Processus principal qui exécute les actions de chaque variable */
proctype Main() {

   local int head_d_x_1 = 1;
   local int head_d_x_2 = 1;
   local int head_d_x_1_2 = 1;
   local int head_d_x_2_2 = 1;
   local int last_x_1 = 0;
   local int last_x_2 = 0;
   local int last_px_1 = 0;
   local int last_px_2 = 0;

   do
      :: atomic{
         Temps(T);
         X1(T, x_1, d_x_1, H_11);
         X2(T, x_2, d_x_2, H_22);
         X11(T, px_1, H_11, head_x_1, head_d_x_1);
         X22(T, px_2, H_22, head_x_2, head_d_x_2);
         Y(T, y, H_y, head_x_1, head_x_2, head_d_x_1, head_d_x_2,
            px_1, px_2, head_x_1_2, head_x_2_2, head_px_1, head_px_2,
            head_d_x_1_2, head_d_x_2_2);
      }
   od
}

/* Initialisation */
init {
   x_1 = 0;
   x_2 = 0;
   px_1 = 0;
   px_2 = 0;
   y = 0;
   d_x_1 = 0;
   d_x_2 = 0;
```

```
    run Main();
}
```

B.2.3 *Résultats*

Les résultats obtenus sont les mêmes qu'en utilisant l'outil TLC. Il n'y a donc pas d'erreur détectée indiquant que la spécification n'est pas satisfiable.

Si on modifie le rythme de mise à jour de la variable x_1 pour qu'il soit semblable à celui de x_2, alors la spécification est faisable. Les modifications en conséquence du code Promela permettent bien à Spin de détecter une erreur représentant une exécution infinie.

B.2.4 *Limite à l'utilisation de Spin*

L'outil Spin semble être plus performant que TLC lorsque le nombre d'état du système augmente. Il est donc possible d'étudier des systèmes ayant une architecture plus complexe. Cependant, comme tout outil de model checking, son efficacité décroît rapidement lorsque le nombre d'états du système augmente

L'implantation du système de transitions fini en Promela, si elle est moins directe qu'en TLA, n'est cependant pas plus compliquée. Cependant comme avec TLC, il faut s'assurer de sa correction et la simplifier selon les propriétés temporelles de la spécification qui ont réellement besoin d'être vérifiées.

Le principal problème de Spin est la difficulté d'exploitation des traces représentant des exécutions de la spécification. Il nous paraît beaucoup plus compliqué d'exploiter celles fournies par cet outil plutôt que celles fournies par TLC.